

A Unified Ordering for Termination Proving

Akihisa Yamada^{a,*}, Keiichirou Kusakari^b, Toshiki Sakabe^a

^a*Graduate School of Information Science, Nagoya University, Japan*

^b*Faculty of Engineering, Gifu University, Japan*

Abstract

We introduce a reduction order called the weighted path order (WPO) that subsumes many existing reduction orders. WPO compares weights of terms as in the Knuth-Bendix order (KBO), while WPO allows weights to be computed by a wide class of interpretations. We investigate summations, polynomials and maximums for such interpretations. We show that KBO is a restricted case of WPO induced by summations, the polynomial order (POLO) is subsumed by WPO induced by polynomials, and the lexicographic path order (LPO) is a restricted case of WPO induced by maximums. By combining these interpretations, we obtain an instance of WPO that unifies KBO, LPO and POLO. In order to fit WPO in the modern dependency pair framework, we further provide a reduction pair based on WPO and partial statuses. As a reduction pair, WPO also subsumes matrix interpretations. We finally present SMT encodings of our techniques, and demonstrate the significance of our work through experiments.

Keywords: Term rewriting, Reduction order, Termination

1. Introduction

Proving *termination of term rewrite systems (TRSs)* is one of the most important tasks in program verification and automated theorem proving, where *reduction orders* play a fundamental role. The classic use of reduction orders in termination proving is illustrated in the following example:

Example 1. Consider the following TRS $\mathcal{R}_{\text{fact}}$:

$$\mathcal{R}_{\text{fact}} := \left\{ \begin{array}{ll} \text{fact}(0) & \rightarrow s(0) \\ \text{fact}(s(x)) & \rightarrow s(x) * \text{fact}(x) \end{array} \right.$$

which defines the factorial function, provided the binary symbol $*$ is defined as multiplication. We can prove termination of $\mathcal{R}_{\text{fact}}$ by finding a reduction order \succ that satisfies the following constraints:

$$\begin{aligned} \text{fact}(0) &\succ s(0) \\ \text{fact}(s(x)) &\succ s(x) * \text{fact}(x) \end{aligned}$$

*Corresponding author

Email addresses: ayamada@trs.cm.is.nagoya-u.ac.jp (Akihisa Yamada), kusakari@gifu-u.ac.jp (Keiichirou Kusakari), sakabe@is.nagoya-u.ac.jp (Toshiki Sakabe)

A number of reduction orders have been proposed, and their efficient implementation is demonstrated by several automatic termination provers such as AProVE [1] or T_TT₂ [2].

One of the most well-known reduction orders is the *lexicographic path order* (LPO) of Kamin and Lévy [3], a variant of the *recursive path order* (RPO) of Dershowitz [4]. LPO is unified with RPO using *status* [5]. Recently, Codish *et al.* [6] proposed an efficient implementation using a SAT solver for termination proving by RPO with status.

The *Knuth-Bendix order* (KBO) [7] is the oldest reduction order. KBO has become a practical alternative in automatic termination checking since Korovin and Voronkov [8] discovered a polynomial-time algorithm for termination proofs with KBO. Zankl *et al.* [9] proposed another implementation method via SAT/SMT encoding, and verified a significant improvement in efficiency over dedicated implementations of the polynomial-time algorithm. However, KBO is disadvantageous compared to LPO when *duplicating* rules (where a variable occurs more often in the right-hand side than in the left-hand side) are considered. Actually, no duplicating rule can be oriented by KBO. To overcome this disadvantage, Middeldorp and Zantema [10] proposed the *generalized KBO* (GKBO), which generalizes weights over algebras that are weakly monotone and *strictly simple*: $f(\dots, x, \dots) > x$. Ludwig and Waldmann proposed another extension of KBO called the *transfinite KBO* (TKBO) [11, 12, 13], which extends the weight function to allow linear polynomials over ordinals. However, proving termination with TKBO involves solving the satisfiability problem of non-linear arithmetic which is undecidable in general. Moreover, TKBO still does not subsume LPO.

The *polynomial order* (POLO) of Lankford [14] interprets each function symbol by a strictly monotone polynomial. Zantema [15] extended the method to algebras and suggested combining the “max” operator with polynomial interpretations (*max-polynomials* in terms of [16]). Fuhs *et al.* proposed an efficient SAT encoding of POLO in [17], and a general version of POLO with max in [16].

The *dependency pair* (DP) *method* of Arts and Giesl [18] significantly enhances the classic approach of reduction orders by analyzing cyclic dependencies between rewrite rules. In the DP method, reduction orders are extended to *reduction pairs* $\langle \succsim, \succ \rangle$, and it suffices if one rule in a recursive dependency is strictly oriented, and other rules are only weakly oriented.

Example 2. Consider again the TRS $\mathcal{R}_{\text{fact}}$. There is one cyclic dependency in $\mathcal{R}_{\text{fact}}$, that is represented by the *dependency pair* $\text{fact}^\sharp(s(x)) \rightarrow \text{fact}^\sharp(p(x))$, where fact^\sharp is a fresh symbol. We can prove termination of $\mathcal{R}_{\text{fact}}$ by finding a reduction pair $\langle \succsim, \succ \rangle$ that satisfies the following constraints:¹

$$\begin{aligned} \text{fact}^\sharp(s(x)) &\succ \text{fact}^\sharp(x) \\ \text{fact}(0) &\succsim s(0) \\ \text{fact}(s(x)) &\succsim s(x) * \text{fact}(x) \end{aligned}$$

One of the typical methods for designing reduction pairs is *argument filtering* [18], which generates reduction pairs from arbitrary reduction orders. Hence,

¹The last two constraints can be removed by considering *usable rules* [18].

reduction orders are still an important subject to study in modern termination proving. Another typical technique is generalizing interpretation methods to *weakly* monotone ones, e.g. allowing 0 coefficients for polynomial interpretations [18]. Endrullis *et al.* [21] extended polynomial interpretations to *matrix interpretations*, and presented their implementation via SAT encoding. More recently, Bofill *et al.* [22] proposed a reduction pair called *RPOLO*, which unifies standard POLO and RPO by choosing either *RPO-like* or *POLO-like* comparison depending on function symbols.

These reduction orders and reduction pairs require different correctness proofs and different implementations. In this paper, we extract the underlying essence of these reduction orders and introduce a general reduction order called the *weighted path order (WPO)*. Technically, WPO is a further generalization of GKBO that relaxes the strict simplicity condition of weights to *weak simplicity*. This relaxation becomes possible by combining the recursive checks of LPO with GKBO. While strict simplicity is so restrictive that GKBO does not even subsume the standard KBO, weak simplicity is so general that WPO subsumes not only KBO but also most of the reduction orders described above (LPO, TKBO, POLO and so on), except for matrix interpretations which are not weakly simple in general.

There exist several earlier works on generalizing existing reduction orders. The *semantic path order (SPO)* of Kamin and Lévy [3] is a generalization of RPO where precedence comparison is generalized to an arbitrary well-founded order on terms. However, to prove termination by SPO users have to ensure monotonicity by themselves, even if the underlying well-founded order is monotone (cf. [23]). On the other hand, monotonicity of WPO is guaranteed. Borralleras *et al.* [23] propose a variant of SPO that ensures monotonicity by using an external monotonic order. As well as LPO or POLO, also WPO can be used as such an external order. The *general path order (GPO)* [24, 25] is a very general framework that many reduction orders are subsumed. Due to the generality, however, implementing GPO seems to be quite challenging. Indeed, we are not aware of any tool that implements GPO.

Instances of WPO are characterized by how weights are computed. In particular, we introduce the following instances of WPO and investigate their relationships with existing reduction orders:

- $\text{WPO}(\text{Sum})$ which uses summations for weight computation. KBO can be obtained as a restricted case of $\text{WPO}(\text{Sum})$, where the *admissibility* condition is enforced, and weights of constants must be greater than 0. $\text{WPO}(\text{Sum})$ is free from these restrictions, and we verify that each extension strictly increases the power of the order.
- $\text{WPO}(\text{Pol})$ which uses monotone polynomial interpretations for weight computation. As a reduction order, POLO is subsumed by $\text{WPO}(\text{Pol})$. TKBO can be obtained as a restricted case of $\text{WPO}(\text{Pol})$, where interpretations are linear polynomials, admissibility is enforced, and interpretations of constants are greater than 0.
- $\text{WPO}(\text{Max})$ which uses maximums for weight computation. LPO can be obtained as a restricted case of $\text{WPO}(\text{Max})$, where the weights of all symbols are fixed to 0. In order to keep the presentation simple, we omit

multiset status and only consider *LPO with status*. Nonetheless, it is easy to extend this result to *RPO with status*.

- $\text{WPO}(\mathcal{MPol})$ which combines polynomials and maximum for interpretation, and its variant $\text{WPO}(\mathcal{MSum})$ whose coefficients are fixed to 1. $\text{WPO}(\mathcal{MSum})$ generalizes KBO and LPO, and $\text{WPO}(\mathcal{MPol})$ moreover subsumes POLO (with max) as a reduction order.

Note that all the instances described above use weakly simple algebras which cannot be used for GKBO.

Next we extend WPO to a reduction pair by incorporating *partial statuses* [26]. This extension further relaxes the weak simplicity condition, and arbitrary weakly monotone interpretations can be used for weight computation. Hence as a reduction pair, WPO also subsumes matrix interpretations, as well as KBO, TKBO, LPO and POLO. Though RPOLO also unifies RPO and POLO, we show that WPO and RPOLO are incomparable in general.² Moreover in practice, WPO brings significant benefit on the problems from the *Termination Problem Data Base (TPDB)* [27], while (the first-order version of) RPOLO does not, as reported in [22].

Finally, we present an efficient implementation using state-of-the-art SMT solvers. By extending [9], we present SMT encoding techniques for the instances of WPO introduced so far. In particular, the orientability problems of $\text{WPO}(\text{Sum})$, $\text{WPO}(\text{Max})$ and $\text{WPO}(\mathcal{MSum})$ are reduced to a satisfiability problem of linear arithmetic, which is known to be decidable. Through experiments in TPDB problems, we also verify the efficiency of our implementation and significance of WPO in practice.

The remainder of this paper is organized as follows: In Section 2 we recall some basic notions of term rewriting and the definitions of existing orders. Section 3 is devoted to WPO as a reduction order. There we present the definition of WPO, and then investigate several instances of WPO and show their relationships with existing orders. In Section 4 we extend WPO to a reduction pair. We present the definition of the reduction pair and a soundness proof in Section 4.1. Then the definition is refined in Section 4.2 and the relationship to existing reduction pairs is shown in Section 4.3. Section 5 presents SMT encodings for the instances of WPO introduced so far, and some implementation issues are discussed in Section 6. In Section 7 we verify the significance of our work through experiments and we conclude in Section 8.

A preliminary version of this paper appeared in [28]. The results for the reduction orders of Section 3 are basically the same as in [28]. The definition of WPO as a reduction pair in Section 4 is new, and hence most of the results in Section 4 are new. The revised version of WPO further subsumes POLO and matrix interpretations as a reduction pair. We also conclude that WPO does not subsume RPOLO by Example 18, which was left open in [28]. The experimental results in Sections 7.2 and 7.3 show a significant improvement due to the new definition of WPO as a reduction pair.

²It is possible to consider an instance of WPO that uses RPOLO for weight computation.

2. Preliminaries

Term rewrite systems (TRSs) model first-order functional programs. We refer the readers to [29, 30] for details on rewriting, and only briefly recall some important notions needed in this paper.

A *signature* \mathcal{F} is a finite set of function symbols associated with an arity. The set of n -ary symbols is denoted by \mathcal{F}_n . A *term* is either a variable $x \in \mathcal{V}$ or of the form $f(s_1, \dots, s_n)$ where $f \in \mathcal{F}_n$ and each s_i is a term. Throughout the paper, we abbreviate a sequence a_1, \dots, a_n by \bar{a}_n . The set of terms constructed from \mathcal{F} and \mathcal{V} is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The set of variables occurring in a term s is denoted by $\text{Var}(s)$, and the number of occurrences of a variable x in s is denoted by $|s|_x$. A TRS is a set \mathcal{R} of pairs of terms called *rewrite rules*. A rewrite rule, written $l \rightarrow r$ where $l \notin \mathcal{V}$ and $\text{Var}(l) \supseteq \text{Var}(r)$, indicates that an instance of l should be rewritten to the corresponding instance of r . The *rewrite relation* $\rightarrow_{\mathcal{R}}$ induced by \mathcal{R} is the least relation which includes \mathcal{R} and is monotonic and stable. Here, a relation \sqsubset on terms is

- *monotonic* iff $s \sqsubset t$ implies $f(\dots, s, \dots) \sqsubset f(\dots, t, \dots)$ for every context $f(\dots, \square, \dots)$, and
- *stable* iff $s \sqsubset t$ implies $s\theta \sqsubset t\theta$ for every substitution θ .

A TRS \mathcal{R} is *terminating* iff no infinite rewrite sequence $s_1 \rightarrow_{\mathcal{R}} s_2 \rightarrow_{\mathcal{R}} \dots$ exists.

2.1. Reduction Orders

A classic method for proving termination is to find a *reduction order*: A *reduction order* is a well-founded order which is monotonic and stable. We say an order \succ *orients* a TRS \mathcal{R} iff $l \succ r$ for every rule $l \rightarrow r \in \mathcal{R}$; in other words, $\mathcal{R} \subseteq \succ$. It is easy to see the following:

Theorem 1. [31] *A TRS is terminating iff it is oriented by a reduction order.* □

Ensuring well-foundedness of a reduction order is often a non-trivial task. The following is a well-known technique of Dershowitz [4] for ensuring well-foundedness based on *Kruskal's tree theorem*. A *simplification order* is a strict order \succ on terms, which is monotonic and stable and satisfies the *subterm property*: $f(\dots, s, \dots) \succ s$.

Theorem 2. [4] *For a finite signature, a simplification order is a reduction order.* □

In the latter, we only consider finite signatures. In the remainder of this section, we recall several existing reduction orders.

2.1.1. Lexicographic Path Order

We consider LPO [3] with quasi-precedence and status; a *quasi-precedence* $\succsim_{\mathcal{F}}$ is a quasi-order (i.e., a reflexive and transitive relation) on \mathcal{F} , whose strict part, denoted by $>_{\mathcal{F}}$, is well-founded. The equivalence part of $\succsim_{\mathcal{F}}$ is denoted by $\sim_{\mathcal{F}}$. A *status function* σ assigns to each function symbol $f \in \mathcal{F}_n$ a permutation $[\bar{i}_n]$ of positions in $\{1, \dots, n\}$. We denote the list $[s_{i_1}, \dots, s_{i_n}]$ by $[\bar{s}_{\sigma(f)}]$ for $\sigma(f) = [\bar{i}_n]$. A strict order \succ (associated with a quasi-order \succsim) is lifted on lists as follows: $[\bar{s}_n] \succ^{\text{lex}} [\bar{y}_m]$ iff there exists $k < n$ s.t. $x_i \succsim y_i$ for each $i \in \{1, \dots, k\}$ and either $k = m$ or $k < m$ and $x_{k+1} \succ y_{k+1}$.

Definition 1. For a quasi-precedence $\succsim_{\mathcal{F}}$, the *lexicographic path order* \succ_{LPO} with status σ is recursively defined as follows: $s = f(\overline{s}_n) \succ_{\text{LPO}} t$ iff

- (a) $\exists i \in \{1, \dots, n\}. s_i \succeq_{\text{LPO}} t$, or
- (b) $t = g(\overline{t}_m), \forall j \in \{1, \dots, m\}. s \succ_{\text{LPO}} t_j$ and either
 - i. $f >_{\mathcal{F}} g$, or
 - ii. $f \sim_{\mathcal{F}} g$ and $[\overline{s}_{\sigma(f)}] \succ_{\text{LPO}}^{\text{lex}} [\overline{t}_{\sigma(g)}]$.

Theorem 3. $[\beta] \succ_{\text{LPO}}$ is a simplification order and hence a reduction order. \square

Example 3. Termination of $\mathcal{R}_{\text{fact}}$ from Example 1 can be shown by LPO. Both rules are oriented by case (b-i) with a precedence s.t. $\text{fact} >_{\mathcal{F}} s$ for the first rule and $\text{fact} >_{\mathcal{F}} *$ for the second rule.

2.1.2. Polynomial Interpretations

We basically follow the abstract definitions of [15, 21]. A *well-founded \mathcal{F} -algebra* \mathcal{A} is a quadruple $\langle A, \succsim, >, \cdot_{\mathcal{A}} \rangle$ of a *carrier set* A , a quasi-order \succsim on A , a well-founded order $>$ on A which is *compatible* with \succsim , i.e., $\succsim \circ > \circ \succsim \subseteq >$, and an *interpretation* $f_{\mathcal{A}} : A^n \rightarrow A$ for each $f \in \mathcal{F}_n$. \mathcal{A} is *strictly (weakly) monotone* iff $a \succsim b$ implies $f_{\mathcal{A}}(\dots, a, \dots) \succsim f_{\mathcal{A}}(\dots, b, \dots)$, and *strictly (weakly) simple* iff $f_{\mathcal{A}}(\dots, a, \dots) \succsim a$ for every $f \in \mathcal{F}$. The relations \succsim and $>$ are extended to terms as follows: $s \succsim_{\mathcal{A}} t$ iff $\hat{\alpha}(s) \succsim \hat{\alpha}(t)$ holds for every assignment $\alpha : \mathcal{V} \rightarrow A$, where $\hat{\alpha} : \mathcal{T} \rightarrow A$ is the homomorphic extension of α . A *polynomial interpretation* \mathcal{Pol} interprets every function symbol $f \in \mathcal{F}$ as a polynomial $f_{\mathcal{Pol}}$. The carrier set of \mathcal{Pol} is $\{a \in \mathbb{N} \mid a \geq w_0\}$ for some $w_0 \in \mathbb{N}$, and the orderings are the standard \geq and $>$ on \mathbb{N} . \mathcal{Pol} induces a reduction order if it is strictly monotone; in other words, all arguments have coefficients at least 1.

Theorem 4. [32, 14] If \mathcal{Pol} is strictly monotone, then $>_{\mathcal{Pol}}$ is a reduction order. \square

Example 4. Termination of $\mathcal{R}_{\text{fact}}$ of Example 1 can be shown by the polynomial interpretation \mathcal{Pol} defined as follows:

$$\begin{aligned} \text{fact}_{\mathcal{Pol}}(x) &= 2x + 2 & 0_{\mathcal{Pol}} &= 0 \\ s_{\mathcal{Pol}}(x) &= 2x + 1 & x *_{\mathcal{Pol}} y &= x + y \end{aligned}$$

The left- and right-hand sides of the rule $\text{fact}(0) \rightarrow s(0)$ are interpreted as 2 and 1, resp., and those of the rule $\text{fact}(s(x)) \rightarrow s(x) * \text{fact}(x)$ are interpreted as $4x + 4$ and $4x + 3$, resp.

2.1.3. Knuth-Bendix Order

KBO [7] is induced by a quasi-precedence and a *weight function* $\langle w, w_0 \rangle$, where $w : \mathcal{F} \rightarrow \mathbb{N}$ and $w_0 \in \mathbb{N}$ s.t. $w(c) \geq w_0$ for every constant $c \in \mathcal{F}_0$. The weight $w(s)$ of a term s is defined as follows:

$$w(s) := \begin{cases} w_0 & \text{if } s \in \mathcal{V} \\ w(f) + \sum_{i=1}^n w(s_i) & \text{if } s = f(\overline{s}_n) \end{cases}$$

The weight function w is said to be *admissible* for $\succsim_{\mathcal{F}}$ iff every unary symbol $f \in \mathcal{F}_1$ with $w(f) = 0$ is *greatest* w.r.t. $\succsim_{\mathcal{F}}$, i.e., $f \succsim_{\mathcal{F}} g$ for every $g \in \mathcal{F}$. In this paper we also consider status for KBO [33].

Definition 2. For a quasi-precedence $\succsim_{\mathcal{F}}$ and a weight function $\langle w, w_0 \rangle$, the *Knuth-Bendix order* \succ_{KBO} with status σ is recursively defined as follows: $s = f(\overline{s}_n) \succ_{\text{KBO}} t$ iff $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either

1. $w(s) > w(t)$, or
2. $w(s) = w(t)$ and either
 - (a) $s = f^k(t)$ and $t \in \mathcal{V}$ for some $k > 0$, or
 - (b) $t = g(\overline{t}_m)$ and either
 - i. $f \succ_{\mathcal{F}} g$, or
 - ii. $f \sim_{\mathcal{F}} g$ and $[\overline{s}_{\sigma(f)}] \succ_{\text{KBO}}^{\text{lex}} [\overline{t}_{\sigma(g)}]$.

Here we follow [9], and the range of w is restricted to \mathbb{N} . According to [8], this does not decrease the power of KBO for finite TRSs. Note that we do not assume $w_0 > 0$ in the definition. This assumption, together with admissibility is required for KBO to be a simplification order. For details of the following result, we refer e.g. to [29, Theorem 5.4.20].

Theorem 5. If $w_0 > 0$ and w is admissible for $\succsim_{\mathcal{F}}$, then \succ_{KBO} induced by $\langle w, w_0 \rangle$ and $\succsim_{\mathcal{F}}$ is a simplification order, and hence a reduction order. \square

The *variable condition* “ $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ ” is often said to be a major disadvantage of KBO. Due to this condition, no duplicating rule can be oriented by KBO.

Example 5. Termination of $\mathcal{R}_{\text{fact}}$ of Example 1 cannot be shown by KBO, since the variable x of the rule $\text{fact}(s(x)) \rightarrow s(x) * \text{fact}(x)$ violates the variable condition.

2.1.4. Transfinite KBO

TKBO [11, 12, 13] extends KBO by introducing a *subterm coefficient function* sc , that assigns a positive integer³ $sc(f, i)$ to each $f \in \mathcal{F}_n$ and $i \in \{1, \dots, n\}$. For a weight function $\langle w, w_0 \rangle$ and a subterm coefficient function sc , the refined weight $w(s)$ is defined as follows:

$$w(s) := \begin{cases} w_0 & \text{if } s \in \mathcal{V} \\ w(f) + \sum_{i=1}^n sc(f, i) \cdot w(s_i) & \text{if } s = f(\overline{s}_n) \end{cases}$$

The *variable coefficient* $\text{vc}(x, s)$ of x in s is defined recursively as follows:

$$\text{vc}(x, s) := \begin{cases} 1 & \text{if } x = s \\ 0 & \text{if } x \neq s \in \mathcal{V} \\ \sum_{i=1}^n sc(f, i) \cdot \text{vc}(x, s_i) & \text{if } s = f(\overline{s}_n) \end{cases}$$

³We do not use *transfinite* coefficients, since they do not add power when finite TRSs are considered [13]. We will still use the acronym TKBO to denote the finite variant.

Then the order \succ_{TKBO} is obtained from Definition 2 by replacing $|\cdot|_x$ by $\text{vc}(x, \cdot)$ and $w(\cdot)$ by its refined version above.

Theorem 6. [11] *If $w_0 > 0$ and w is admissible for $\succsim_{\mathcal{F}}$, then \succ_{TKBO} is a simplification order and hence a reduction order.* \square

Example 6. Consider again the TRS $\mathcal{R}_{\text{fact}}$ of Example 1. Consider the weight function w defined as follows:

$$w(0) = 1 \quad w(s) = 0 \quad w(\text{fact}) = 1 \quad w(*) = 0$$

and the subterm coefficient function sc defined as follows:

$$sc(s, 1) = sc(\text{fact}, 1) = 2 \quad sc(*, 1) = sc(*, 2) = 1$$

Finally, consider an admissible precedence s.t. $s >_{\mathcal{F}} \text{fact} >_{\mathcal{F}} *$. The first rule $\text{fact}(0) \rightarrow s(0)$ of $\mathcal{R}_{\text{fact}}$ is oriented by case (1). For both sides of the second rule $\text{fact}(s(x)) \rightarrow s(x) * \text{fact}(x)$, the variable coefficient of x is 4 and the weight is 3. Hence, the rule is oriented by case (2b-i).

2.1.5. Generalized Knuth-Bendix Order

GKBO [10] uses a weakly monotone and *strictly* simple algebra for weight computation. In the following version of GKBO, we extend [10] with quasi-order $\succsim_{\mathcal{A}}$ and quasi-precedence $\succsim_{\mathcal{F}}$, and omit *multiset status*.

Definition 3. For a quasi-precedence $\succsim_{\mathcal{F}}$ and a well-founded \mathcal{F} -algebra \mathcal{A} , the *generalized Knuth-Bendix order* \succ_{GKBO} is recursively defined as follows: $s = f(\overline{s}_n) \succ_{\text{GKBO}} t$ iff

1. $s >_{\mathcal{A}} t$, or
2. $s \succsim_{\mathcal{A}} t = g(\overline{t}_m)$ and either
 - i. $f >_{\mathcal{F}} g$, or
 - ii. $f \sim_{\mathcal{F}} g$ and $[\overline{s}_{\sigma(f)}] \succ_{\text{GKBO}}^{\text{lex}} [\overline{t}_{\sigma(g)}]$.

Theorem 7. [10] *If \mathcal{A} is weakly monotone and strictly simple, then \succ_{GKBO} is a simplification order and hence a reduction order.* \square

One of the most important advantage of GKBO is that it admits weakly monotone interpretations such as \max .

Example 7. Termination of $\mathcal{R}_{\text{fact}}$ of Example 1 can be shown by GKBO induced by an algebra \mathcal{A} on \mathbb{N} with interpretation s.t.

$$\begin{aligned} \text{fact}_{\mathcal{A}}(x) &= x + 2 & 0_{\mathcal{A}} &= 0 \\ s_{\mathcal{A}}(x) &= x + 1 & x *_{\mathcal{A}} y &= \max\{x, y\} + 1 \end{aligned}$$

and a precedence s.t. $\text{fact} >_{\mathcal{F}} *$. The first rule $\text{fact}(0) \rightarrow s(0)$ is oriented by case (1). The second rule $\text{fact}(s(x)) \rightarrow s(x) * \text{fact}(x)$ is oriented by case (2i), since $x + 3 \geq \max\{x + 1, x + 2\} + 1$.

Note that the strict simplicity condition is crucial for GKBO to be well-founded. If we modify the interpretation of the above example by $x *_{\mathcal{A}} y = \max\{x, y\}$, then GKBO will admit the following infinite sequence:

$$\text{fact}(0) \succ_{\text{GKBO}} 0 * \text{fact}(0) \succ_{\text{GKBO}} 0 * (0 * \text{fact}(0)) \succ_{\text{GKBO}} \dots$$

2.2. The Dependency Pair Framework and Reduction Pairs

The *dependency pair (DP) method* [18] significantly enhances the classical method of reduction orders by analyzing dependencies between rewrite rules. We briefly recall the essential notions for its successor, the *DP framework* [19, 34, 20].

Let \mathcal{R} be a TRS over a signature \mathcal{F} . The *root symbol* of a term $s = f(\overline{s}_n)$ is f and denoted by $\text{root}(s)$. The set of *defined symbols* w.r.t. \mathcal{R} is defined as $\mathcal{D} := \{\text{root}(l) \mid l \rightarrow r \in \mathcal{R}\}$. For each $f \in \mathcal{D}$, the signature \mathcal{F} is extended by a fresh *marked symbol* $f^\#$ having the same arity as f . For $s = f(\overline{s}_n)$ with $f \in \mathcal{D}$, the term $f^\#(\overline{s}_n)$ is denoted by $s^\#$. The set of *dependency pairs* for \mathcal{R} is defined as $\text{DP}(\mathcal{R}) := \{l^\# \rightarrow t^\# \mid l \rightarrow r \in \mathcal{R}, t \text{ is a subterm of } r, \text{root}(t) \in \mathcal{D}\}$. A *DP problem* is a pair $\langle \mathcal{P}, \mathcal{R} \rangle$ of a TRS \mathcal{R} and a set \mathcal{P} of dependency pairs for \mathcal{R} . A DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ is *finite* iff $\rightarrow_{\mathcal{P}} \cdot \rightarrow_{\mathcal{R}}^*$ is well-founded, where \mathcal{P} is viewed as a TRS. The main result of the DP framework is the following:

Theorem 8. [18, 20] *A TRS \mathcal{R} is terminating iff the DP problem $\langle \text{DP}(\mathcal{R}), \mathcal{R} \rangle$ is finite.* \square

Finiteness of a DP problem is proved by *DP processors*: A sound *DP processor* gets a DP problem as input and outputs a set of (hopefully simpler) DP problems s.t. the input problem is finite if all the output problems are finite. Among other DP processors for transforming or simplifying DP problems (cf. [20] for a summary), we recall the most important one: A *reduction pair* $\langle \succsim, \succ \rangle$ is a pair of relations on terms s.t. \succsim is a monotonic and stable quasi-order, and \succ is a well-founded stable order which is *compatible* with \succsim , i.e., $\succsim \circ \succ \circ \succsim \subseteq \succ$.

Theorem 9. [18, 34, 19, 20] *Let $\langle \succsim, \succ \rangle$ be a reduction pair s.t. $\mathcal{P} \cup \mathcal{R} \subseteq \succsim$ and $\mathcal{P}' \subseteq \succ$. Then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound.* \square

2.2.1. Weakly Monotone Interpretations

To define a reduction pair by polynomial interpretations, an interpretation \mathcal{Pol} need not be strictly monotone but only weakly monotone; in other words, 0 coefficients are allowed.

Theorem 10. [18] *If \mathcal{Pol} is weakly monotone, then $\langle \geq_{\mathcal{Pol}}, >_{\mathcal{Pol}} \rangle$ forms a reduction pair.* \square

Endrullis *et al.* [21] extend linear polynomial interpretations to *matrix interpretations*.

Definition 4. Given a fixed *dimension* $d \in \mathbb{N}$, the well-founded algebra \mathcal{Mat} consists of the carrier set \mathbb{N}^d and the strict and quasi-orders on \mathbb{N}^d defined as follows:

$$\begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} \succsim \begin{pmatrix} u_1 \\ \vdots \\ u_d \end{pmatrix} \stackrel{\text{def}}{\iff} v_1 \geq u_1 \text{ and } v_j \geq u_j \text{ for all } j \in \{2, \dots, d\}$$

The interpretation in \mathcal{Mat} is induced by a function \mathbf{w} that assigns a d -dimension vector $\mathbf{w}(f)$ to each $f \in \mathcal{F}$, and a function SC that assigns a $d \times d$ matrix $SC(f, i)$

to each $f \in \mathcal{F}_n$ and $i \in \{1, \dots, n\}$. It is defined as follows:

$$f_{\text{Mat}}(\overline{\mathbf{x}}_n) = \mathbf{w}(f) + \sum_{i=1}^n SC(f, i) \cdot \mathbf{x}_i$$

The i -th row and j -th column element of a matrix M is denoted by $M^{i,j}$.

Theorem 11. [21] *For a matrix interpretation Mat , $\langle \succsim_{\text{Mat}}, \succ_{\text{Mat}} \rangle$ forms a reduction pair.* \square

2.2.2. Argument Filtering

Argument filtering [18, 35] is a typical technique to design a reduction pair from a reduction order: An *argument filter* π maps each $f \in \mathcal{F}_n$ to either a position $i \in \{1, \dots, n\}$ or a list $[\overline{i}_m]$ of positions s.t. $1 \leq i_1 < \dots < i_m \leq n$. The signature \mathcal{F}^π consists of every $f \in \mathcal{F}$ s.t. $\pi(f) = [\overline{i}_m]$, and the arity of f is m in \mathcal{F}^π . An argument filter π induces a mapping $\pi : \mathcal{T}(\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{T}(\mathcal{F}^\pi, \mathcal{V})$ as follows:

$$\pi(s) := \begin{cases} s & \text{if } s \in \mathcal{V} \\ \pi(s_i) & \text{if } s = f(\overline{s}_n), \pi(f) = i \\ f(\pi(s_{i_1}), \dots, \pi(s_{i_m})) & \text{if } s = f(\overline{s}_n), \pi(f) = [\overline{i}_m] \end{cases}$$

For an argument filter π and a reduction order \succ on $\mathcal{T}(\mathcal{F}^\pi, \mathcal{V})$, the relations \succsim^π and \succ^π on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ are defined as follows: $s \succsim^\pi t$ iff $\pi(s) \succeq \pi(t)$, and $s \succ^\pi t$ iff $\pi(s) \succ \pi(t)$.

Theorem 12. [18] *For a reduction order \succ and an argument filter π , $\langle \succsim^\pi, \succ^\pi \rangle$ forms a reduction pair.* \square

The effect of argument filtering is especially apparent for KBO; it relaxes the variable condition.

Example 8. By applying an argument filter π s.t. $\pi(*) = 1$ for the constraints in Example 2, we obtain the following constraints:

$$\text{fact}(0) \succsim s(0) \quad \text{fact}(s(x)) \succsim s(x) \quad \text{fact}^\sharp(s(x)) \succ \text{fact}^\sharp(x)$$

The first constraint can be satisfied by KBO with e.g. $w(\text{fact}) > w(s)$. The other constraints are satisfied by any instance of KBO.

3. WPO as a Reduction Order

In this section, we introduce a reduction order called the *weighted path order* (WPO) that further generalizes GKBO by weakening the simplicity condition on algebras. After showing some properties for the order, we then introduce several instances of WPO by fixing algebras. We investigate relationships between these instances of WPO and existing reduction orders and show the potential of WPO.

We first introduce the definition of WPO. In order to admit algebras that are not strictly but only weakly simple, we employ the recursive checks which ensure that LPO is a simplification order.

Definition 5 (WPO). For a quasi-precedence $\succsim_{\mathcal{F}}$, a well-founded algebra \mathcal{A} and a status σ , the *weighted path order* $\succ_{\text{WPO}(\mathcal{A}, \sigma)}$ is defined as follows: $s = f(\overline{s}_n) \succ_{\text{WPO}(\mathcal{A}, \sigma)} t$ iff

1. $s >_{\mathcal{A}} t$, or
2. $s \succsim_{\mathcal{A}} t$ and
 - (a) $\exists i \in \{1, \dots, n\}. s_i \succeq_{\text{WPO}(\mathcal{A}, \sigma)} t$, or
 - (b) $t = g(\overline{t}_m), \forall j \in \{1, \dots, m\}. s \succ_{\text{WPO}(\mathcal{A}, \sigma)} t_j$ and either
 - i. $f >_{\mathcal{F}} g$ or
 - ii. $f \sim_{\mathcal{F}} g$ and $[\overline{s}_{\sigma(f)}] \succ_{\text{WPO}(\mathcal{A}, \sigma)}^{\text{lex}} [\overline{t}_{\sigma(g)}]$.

We abbreviate $\text{WPO}(\mathcal{A}, \sigma)$ by $\text{WPO}(\mathcal{A})$ and WPO when no confusion arises.

Case (1) and the precondition of case (2) are the same as GKBO. Case (2a) and the precondition of case (2b) are the recursive checks that correspond to (a) and (b) of LPO. Note that here we may restrict i in (2a) and j in (2b) to positions s.t. $f(\dots, x_i, \dots) \not>_{\mathcal{A}} x_i$, since otherwise we have $s >_{\mathcal{A}} t$, which is covered by (1). Cases (2b-i) and (2b-ii) are common among WPO, GKBO and LPO. In the appendix we prove the following soundness result:

Theorem 13. *If \mathcal{A} is weakly monotone and weakly simple, then \succ_{WPO} is a simplification order and hence a reduction order.*

Note that Theorem 13 gives an alternative proof for the following result of Zantema [15]:

Theorem 14. [15] *If a TRS \mathcal{R} is oriented by $>_{\mathcal{A}}$ for a weakly monotone and weakly simple algebra \mathcal{A} , then \mathcal{R} is simply terminating, i.e., its termination is shown by a simplification order.*

Proof. Since $>_{\mathcal{A}} \subseteq \succ_{\text{WPO}}$, \mathcal{R} is oriented by the simplification order \succ_{WPO} . \square

Moreover, we can verify that WPO is a generalization of GKBO.

Theorem 15. *If \mathcal{A} is strictly simple, then $\succ_{\text{GKBO}} = \succ_{\text{WPO}}$.*

Proof. The condition $s_i \succeq_{\text{WPO}} t$ of case (2a) may be dropped, since in that case we have $s >_{\mathcal{A}} s_i \succsim_{\mathcal{A}} t$ by the assumption. Analogously, the condition $s \succ_{\text{WPO}} t_j$ of case (2b) always holds, since $s \succsim_{\mathcal{A}} t >_{\mathcal{A}} t_j$. Hence, case (2a) and the condition in (2b) can be ignored, and the definition of WPO becomes equivalent to that of GKBO. \square

The relaxation of strict simplicity to weak simplicity is an important step. While GKBO does not even subsume the standard KBO, WPO subsumes not only KBO but also many other reduction orders. In the remainder of this section, we investigate several instances of WPO.

3.1. $WPO(\mathcal{S}um)$

The first instance $WPO(\mathcal{S}um)$ is induced by an algebra $\mathcal{S}um$, which interprets function symbols as the summation operator \sum . We obtain KBO as a restricted case of $WPO(\mathcal{S}um)$. We design the algebra $\mathcal{S}um$ from a weight function $\langle w, w_0 \rangle$, so that $\succ_{WPO(\mathcal{S}um)} = \succ_{KBO}$ when $w_0 > 0$ and admissibility is satisfied.

Definition 6. The \mathcal{F} -algebra $\mathcal{S}um$ induced by a weight function $\langle w, w_0 \rangle$ consists of the carrier set $\{a \in \mathbb{N} \mid a \geq w_0\}$ and the interpretation which is defined as follows:

$$f_{\mathcal{S}um}(\bar{a}_n) = w(f) + \sum_{i=1}^n a_i$$

If $w_0 > 0$ is satisfied, we also write $\mathcal{S}um^+$ for $\mathcal{S}um$.

Obviously, $\mathcal{S}um$ is strictly (and hence weakly) monotone and weakly simple. We obtain the following as a corollary of Theorem 13:

Corollary 1. $\succ_{WPO(\mathcal{S}um)}$ is a reduction order. □

Now let us prove that \succ_{KBO} is obtained as a special case of $\succ_{WPO(\mathcal{S}um^+)}$. The following lemma verifies that $\mathcal{S}um$ indeed works as the weight of KBO.

Lemma 1. $s \succeq_{\mathcal{S}um} t$ iff $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and $w(s) \succeq w(t)$.

Proof. The “if” direction is easy. For the “only-if” direction, suppose $s \succeq_{\mathcal{S}um} t$. Define the assignment α_0 which maps all variables to w_0 . We have $\hat{\alpha}_0(s) \succeq \hat{\alpha}_0(t)$, that is $w(s) \succeq w(t)$. Furthermore, define the assignment α_x which maps x to $w(s) + w_0$ and other variables to w_0 . We have $\hat{\alpha}_x(s) \succeq \hat{\alpha}_x(t)$, which implies $w(s) + |s|_x \cdot w(s) \succeq w(t) + |t|_x \cdot w(s)$. Here, $|s|_x < |t|_x$ cannot hold since $w(t) \geq 0$. We conclude $|s|_x \geq |t|_x$. □

Theorem 16. If $w_0 > 0$ and w is admissible for $\succ_{\mathcal{F}}$, then $\succ_{WPO(\mathcal{S}um)} = \succ_{KBO}$.

Proof. For arbitrary terms $s = f(\bar{s}_n)$ and t , we show $s \succ_{WPO(\mathcal{S}um)} t$ iff $s \succ_{KBO} t$ by induction on $|s| + |t|$. Because of the admissibility assumption, we may assume that \succ_{KBO} is a simplification order.

- Suppose $s \succ_{KBO} t$. If $w(s) > w(t)$, then we have $s \succ_{\mathcal{S}um} t$ by Lemma 1 and $s \succ_{WPO(\mathcal{S}um)} t$ by (1) of Definition 5. Let us consider that $w(s) = w(t)$.
 - Suppose $s = f^k(t)$ and $t \in \mathcal{V}$ for some $k > 0$. Since $w(s) = w(t)$, $w(f) = 0$. If $k = 1$, then we are done by case (2a). Otherwise $f^{k-1}(t) \succ_{KBO} t$ by case (2a) of Definition 2. By the induction hypothesis we get $f^{k-1}(t) \succ_{WPO(\mathcal{S}um)} t$, and hence case (2a) of Definition 5 applies.
 - Suppose $t = g(\bar{t}_m)$ and case (2b-i) or (2b-ii) applies. For all $j \in \{1, \dots, m\}$, we have $t \succ_{KBO} t_j$ by the subterm property of \succ_{KBO} , and we get $s \succ_{KBO} t_j$ by the transitivity. By the induction hypothesis, $s \succ_{WPO(\mathcal{S}um)} t_j$. Hence, the side condition in (2b) of Definition 5 is satisfied, and subcase (2b-i) or (2b-ii) applies.

- Suppose $s \succ_{\text{WPO}(\text{Sum})} t$. If $s \succ_{\text{Sum}} t$, then $w(s) > w(t)$ by Lemma 1 and $s \succ_{\text{KBO}} t$ by (1) of Definition 2. Otherwise we get $w(s) = w(t)$ by Lemma 1.
 - Suppose $s_i \succeq_{\text{WPO}(\text{Sum})} t$ for some $i \in \{1, \dots, n\}$. By the induction hypothesis, we have $s_i \succeq_{\text{KBO}} t$. The subterm property of \succ_{KBO} ensures $s \succ_{\text{KBO}} s_i$. Hence by transitivity, we get $s \succ_{\text{KBO}} t$.
 - Suppose $t = g(\bar{t}_m)$. If $f \succ_{\mathcal{F}} g$, then case (2b-i) of Definition 2 applies. If $f = g$ and $[\bar{s}_n] \succ_{\text{WPO}(\text{Sum})}^{\text{lex}} [\bar{t}_m]$, then by the induction hypothesis we get $[\bar{s}_n] \succ_{\text{KBO}}^{\text{lex}} [\bar{t}_m]$, and hence case (2b-ii) applies. \square

Note that we need neither admissibility nor $w_0 > 0$ in Corollary 1. Let us see that removal of these conditions is indeed advantageous. The following example illustrates that $\text{WPO}(\text{Sum}^+)$ properly extends KBO because admissibility is relaxed.

Example 9. Consider the following TRS \mathcal{R}_1 :

$$\mathcal{R}_1 := \begin{cases} f(g(x)) \rightarrow g(f(f(x))) \\ f(h(x)) \rightarrow h(h(f(x))) \end{cases}$$

The first rule is oriented from right to left by LPO in any precedence. The second rule is oriented from right to left by KBO, since $w(h) > 0$ implies increase in weights and $w(h) = 0$ implies $h \succeq_{\mathcal{F}} f$ by the admissibility. On the other hand, $\text{WPO}(\text{Sum}^+)$ with precedence $f \succ_{\mathcal{F}} g$, $f \succ_{\mathcal{F}} h$ and $w(g) > w(f) = w(h) = 0$ orients all the rules. Hence, \mathcal{R}_1 is orientable by $\text{WPO}(\text{Sum}^+)$, but not by KBO or LPO. Note that there is no need to consider a status for \mathcal{R}_1 , since all symbols are unary.

Moreover, allowing $w_0 = 0$ is also a proper enhancement.

Example 10. Consider the following TRS \mathcal{R}_2 :

$$\mathcal{R}_2 := \begin{cases} f(a, b) \rightarrow f(b, f(b, a)) \\ f(a, f(b, x)) \rightarrow f(x, f(b, b)) \end{cases}$$

The first rule cannot be oriented by KBO or $\text{WPO}(\text{Sum}^+)$, since $w(b) = 0$ is required. The second rule is not orientable by LPO no matter how one chooses σ . On the other hand, $\text{WPO}(\text{Sum})$ with $w(a) > w(b) = w(f) = 0$, $a \succ_{\mathcal{F}} b$ and $\sigma(f) = [1, 2]$ orients both rules. Hence, \mathcal{R}_2 is orientable by $\text{WPO}(\text{Sum})$ with $w_0 = 0$, but not by LPO, KBO, or $\text{WPO}(\text{Sum}^+)$.

3.2. $\text{WPO}(\text{Pol})$

Next we consider generalizing $\text{WPO}(\text{Sum})$ using monotone polynomial interpretations. According to Zantema [15, Proposition 4], every monotone interpretation on a totally ordered set is weakly simple. Hence a monotone polynomial interpretation Pol is weakly simple and we obtain the following:

Corollary 2. *If Pol is strictly monotone, then $\succ_{\text{WPO}(\text{Pol})}$ is a reduction order.* \square

Trivially, POLO is subsumed by $\text{WPO}(\mathcal{P}ol)$ as a reduction order. More precisely, the following relation holds:

Theorem 17. $\succ_{\mathcal{P}ol} \subseteq \succ_{\text{WPO}(\mathcal{P}ol)}$. \square

In the remainder of this paper, we consider an algebra $\mathcal{P}ol$ that consists of linear polynomial interpretations induced by a weight function $\langle w, w_0 \rangle$ and a subterm coefficient function sc , which is defined as follows:

$$f_{\mathcal{P}ol}(\bar{a}_n) := w(f) + \sum_{i=1}^n sc(f, i) \cdot a_i$$

Analogous to Theorem 16, we also obtain the following:

Theorem 18. If $w_0 > 0$ and w is admissible for $\succsim_{\mathcal{F}}$, then $\succ_{\text{WPO}(\mathcal{P}ol)} = \succ_{\text{TKBO}}$. \square

Moreover, we can verify that $\text{WPO}(\mathcal{P}ol)$ strictly enhances both POLO and TKBO. More precisely, we show that both POLO and TKBO do not subsume even $\text{WPO}(\text{Sum})$.

Example 11. POLO cannot orient the first rule of \mathcal{R}_1 :

$$l_1 = f(g(x)) \rightarrow g(f(f(x))) = r_1$$

since it is not ω -terminating [31]. Suppose that \mathcal{R}_1 is oriented by TKBO. For the first rule, we need

$$\text{vc}(x, l_1) = sc(f, 1) \cdot sc(g, 1) \geq sc(g, 1) \cdot sc(f, 1)^2 = \text{vc}(x, r_1)$$

Hence $sc(f, 1) = 1$. Moreover,

$$\begin{aligned} w(l_1) &= w(f) + w(g) + sc(g, 1) \cdot w_0 \\ &\geq w(g) + sc(g, 1) \cdot (2 \cdot w(f) + w_0) = w(r_1) \end{aligned}$$

Hence $w(f) = 0$. Analogously, for the second rule of \mathcal{R}_1 :

$$l_2 = f(h(x)) \rightarrow h(h(f(x))) = r_2$$

we need $sc(h, 1) = 1$ and $w(h) = 0$. Hence $w(l_2) = w(r_2)$. The admissibility imposes $f \sim_{\mathcal{F}} h$, and thus the rule is oriented only from right to left.

Note also that it is not possible to orient one of the rules in \mathcal{R}_1 by $\succ_{\mathcal{P}ol}$ and the other by $\geq_{\mathcal{P}ol}$. Thus, together with the discussion in Example 9, we conclude that \mathcal{R}_1 cannot be oriented by any lexicographic composition of POLO, (T)KBO, and LPO.

3.3. $\text{WPO}(\text{Max})$

Note that $\mathcal{P}ol$ is strictly monotone. WPO also admits *weakly* monotone interpretations; a typical example is max .⁴ Let us consider an instance of WPO using max for interpretation.

⁴ Note that weakly monotone polynomials with 0 coefficients are not weakly simple, and hence cannot be applied for WPO of this section. We consider such polynomials in Section 4.

Definition 7. A *subterm penalty function* sp is a mapping s.t. $sp(f, i) \in \mathbb{N}$ is defined for each $f \in \mathcal{F}_n$ and $i \in \{1, \dots, n\}$. A weight function $\langle w, w_0 \rangle$ and sp induce the \mathcal{F} -algebra \mathcal{Max} , which consists of the carrier set $\{a \in \mathbb{N} \mid a \geq w_0\}$ and interpretations given by:

$$f_{\mathcal{Max}}(\bar{a}_n) := \max \left(w(f), \max_{i=1}^n (sp(f, i) + a_i) \right)$$

Lemma 2. \mathcal{Max} is weakly monotone and weakly simple.

Proof. Weak simplicity is obvious from the fact that $\max(\dots, a, \dots) \geq a$. For weak monotonicity, suppose $a > b$ and let us show

$$a' = f_{\mathcal{Max}}(\bar{c}_k, a, \bar{d}_l) \geq f_{\mathcal{Max}}(\bar{c}_k, b, \bar{d}_l) = b'$$

To this end, let $c = f_{\mathcal{Max}}(\bar{c}_k, 0, \bar{d}_l)$. If $c \geq sp(f, k+1) + a$, then $a' = b' = c$. Otherwise, we have $a' = sp(f, k+1) + a$ and either $a' > sp(f, k+1) + b = b' > c$ or $a' > b' = c$. \square

Note that \mathcal{Max} can be considered as the 1-dimensional variant of *arctic interpretations* [36]. The weak monotonicity of \mathcal{Max} is also shown there.

Corollary 3. $\succ_{\text{WPO}(\mathcal{Max})}$ is a reduction order. \square

Now we show that LPO is obtained as a restricted case of $\text{WPO}(\mathcal{Max})$.

Theorem 19. If $w_0 = 0$, $w(f) = 0$ and $sp(f, i) = 0$ for all $f \in \mathcal{F}_n$ and $i \in \{1, \dots, n\}$, then $\succ_{\text{LPO}} = \succ_{\text{WPO}(\mathcal{Max})}$.

Proof. From the assumptions, $s >_{\mathcal{Max}} t$ never holds. Hence, case (1) of Definition 5 can be ignored. Moreover, $s \geq_{\mathcal{Max}} t$ is equivalent to $\text{Var}(s) \supseteq \text{Var}(t)$. One can easily verify the latter holds whenever $s \succ_{\text{LPO}} t$, using the fact that $s \not\prec_{\text{LPO}} x$ for $x \notin \text{Var}(s)$. Hence, the condition of case (2) can be ignored and Definition 1 and Definition 5 become equivalent. \square

The following example illustrates that $\text{WPO}(\mathcal{Max})$ properly enhances LPO.

Example 12. Consider the following TRS \mathcal{R}_3 :

$$\mathcal{R}_3 := \begin{cases} f(x, y) \rightarrow g(x) \\ f(g(x), y) \rightarrow f(x, g(x)) \\ f(x, g(y)) \rightarrow f(y, y) \end{cases}$$

To orient the first two rules by LPO, we need $f >_{\mathcal{F}} g$ and $\sigma(f) = [1, 2]$. LPO cannot orient the third rule by this precedence and status, while $sp(g, 1) > sp(f, 1) = 0$ suffices for $\text{WPO}(\mathcal{Max})$. Since the last two rules are duplicating, KBO or $\text{WPO}(\text{Sum})$ cannot apply for \mathcal{R}_3 .

However, $\text{WPO}(\mathcal{Max})$ covers neither $\text{WPO}(\text{Sum})$ nor even KBO. In the next section, we consider unifying $\text{WPO}(\text{Sum})$ and $\text{WPO}(\mathcal{Max})$ to cover both KBO and LPO.

3.4. $WPO(\mathcal{MPol})$ and $WPO(\mathcal{MSum})$

Now we consider unifying $WPO(\mathcal{Max})$ and $WPO(\mathcal{Pol})$. To this end, we introduce the *weight status* to choose a polynomial or max for each function symbol.

Definition 8. A *weight status function* is a mapping ws which maps each function symbol f either to the symbol **pol** or to the symbol **max**. The \mathcal{F} -algebra \mathcal{MPol} consists of the carrier set $\{a \in \mathbb{N} \mid a \geq w_0\}$ and the interpretation which is defined as follows:

$$f_{\mathcal{MPol}}(\bar{a}_n) := \begin{cases} w(f) + \sum_{i=1}^n sc(f, i) \cdot a_i & \text{if } ws(f) = \text{pol} \\ \max\left(w(f), \max_{i=1}^n (sp(f, i) + a_i)\right) & \text{if } ws(f) = \text{max} \end{cases}$$

We denote \mathcal{MPol} by \mathcal{MSum} if coefficients are at most 1.

Corollary 4. $WPO(\mathcal{MPol})$ is a reduction order. \square

Trivially, $WPO(\mathcal{MSum})$ subsumes both $WPO(\mathcal{Sum})$ and $WPO(\mathcal{Max})$. Hence, we obtain the following more interesting result:

Theorem 20. $WPO(\mathcal{MSum})$ subsumes both LPO and KBO. \square

As far as we know, this is the first reduction order that unifies LPO and KBO. The following example illustrates that $WPO(\mathcal{MSum})$ is strictly stronger than the union of $WPO(\mathcal{Sum})$ and $WPO(\mathcal{Max})$.

Example 13. Consider the following TRS \mathcal{R}_4 :

$$\mathcal{R}_4 := \begin{cases} f(f(x, y), z) \rightarrow f(x, f(y, z)) \\ g(f(a, x), b) \rightarrow g(f(x, b), x) \end{cases}$$

If $ws(f) = \text{max}$, then the first rule requires $sp(f, 2) = 0$. Under this restriction the second rule cannot be oriented. If $ws(f) = \text{pol}$, then the first rule is oriented iff $sc(f, 1) = sc(f, 2) = 1$ and $\sigma(f) = [1, 2]$. On the other hand, the duplicating variable x in the second rule requires $ws(g) = \text{max}$. Hence, \mathcal{R}_4 is orientable by $WPO(\mathcal{MSum})$ only if $ws(f) = \text{pol}$ and $ws(g) = \text{max}$.

The following example suggests that $WPO(\mathcal{MSum})$ advances the state-of-the-art of automated termination proving.

Example 14. The most powerful termination provers including AProVE 2013 and T_T2 1.11 fail to prove termination of the following TRS \mathcal{R}_5 :

$$\mathcal{R}_5 := \begin{cases} f(g(g(x, a), g(b, y))) \rightarrow f(g(g(h(x, x), b), g(y, a))) \\ g(x, y) \rightarrow x \\ h(x, h(y, z)) \rightarrow y \end{cases}$$

We show that $WPO(\mathcal{MSum})$ with $ws(g) = \text{pol}$, $ws(h) = \text{max}$, $w(a) > w(b)$, $w(h) = sp(h, 1) = sp(h, 2) = 0$ and $\sigma(f) = \sigma(g) = [1, 2]$ orients all the rules. For the first rule, applying case (2b-ii) twice it yields orienting $g(x, a) \succ_{WPO(\mathcal{MSum})} g(h(x, x), b)$ where case (1) applies. The other rules are trivially oriented.

In general, matrix interpretations cannot be combined with WPO, since a matrix interpretation is often not weakly simple. Consider the following TRS from [21]:

$$\mathcal{R}_6 := \{f(f(x)) \rightarrow f(g(f(x)))\}$$

which is shown terminating by the following matrix interpretation \mathcal{Mat} s.t.

$$f_{\mathcal{Mat}}(\vec{x}) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \cdot \vec{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad g_{\mathcal{Mat}}(\vec{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \vec{x}$$

However, $g_{\mathcal{Mat}}$ is not weakly simple. For example,

$$g_{\mathcal{Mat}}\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \not\geq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Hence to unify the matrix interpretation with WPO, we have to further relax the weak simplicity condition. This is achieved in the next section by extending WPO to a reduction pair.

4. WPO as a Reduction Pair

In this section, we extend WPO to a reduction pair. First we introduce the basic definition and prove its soundness, and then we present two refinements. Afterwards we introduce some instances of WPO as reduction pairs and investigate relationships with existing reduction pairs. In particular, matrix interpretations are also subsumed by WPO as a reduction pair.

4.1. WPO with Partial Status

In the preliminary version of this paper [28], we simply applied argument filtering to obtain the reduction pair $\langle \succsim_{\text{WPO}}^\pi, \succ_{\text{WPO}}^\pi \rangle$ from WPO. In this paper, we fully revise this approach and directly define a reduction pair by incorporating *partial statuses* [26] into WPO. A partial status is a generalization of status that admits *non-permutations*.

Definition 9. A *partial status function* σ is a mapping that assigns to each n -ary symbol f a list $[\vec{i}_m]$ of (distinct) positions in $\{1, \dots, n\}$. We also view $\sigma(f)$ as the set $\{\vec{i}_m\}$ for $\sigma(f) = [\vec{i}_m]$. A well-founded algebra \mathcal{A} is (*weakly*) *simple* w.r.t. σ iff $f_{\mathcal{A}}$ is (weakly) simple in its i -th argument for every $f \in \mathcal{F}$ and $i \in \sigma(f)$.

Note that here $\sigma(f)$ need not be a permutation, since some positions may be ignored. If every $\sigma(f)$ is a permutation, then we say that σ is *total*. Conversely if $\sigma(f) = []$ for every f , then we call σ the *empty* status.

Definition 10 (WPO with Partial Status). Let \mathcal{A} be a well-founded algebra and σ a partial status. The pair $\langle \succsim_{\text{WPO}(\mathcal{A}, \sigma)}, \succ_{\text{WPO}(\mathcal{A}, \sigma)} \rangle$ of relations is defined mutually recursively as follows: $x \succsim_{\text{WPO}(\mathcal{A}, \sigma)} x$, and $s = f(\vec{s}_n) \succsim_{\text{WPO}(\mathcal{A}, \sigma)} t$ iff

1. $s \succ_{\mathcal{A}} t$, or
2. $s \succsim_{\mathcal{A}} t$ and

- (a) $\exists i \in \sigma(f). s_i \succsim_{\text{WPO}(\mathcal{A}, \sigma)} t$, or
- (b) $t = g(\bar{t}_m), \forall j \in \sigma(g). s \succ_{\text{WPO}(\mathcal{A}, \sigma)} t_j$ and either
 - i. $f \succ_{\mathcal{F}} g$ or
 - ii. $f \sim_{\mathcal{F}} g$ and $[\bar{s}_{\sigma(f)}] \succsim_{\text{WPO}(\mathcal{A}, \sigma)}^{\text{lex}} [\bar{t}_{\sigma(g)}]$.

In the appendix we prove that the pair $\langle \succsim_{\text{WPO}}, \succ_{\text{WPO}} \rangle$ is indeed a reduction pair.

The effect of a partial status has similarity with that of combining argument filtering and a standard total status. Indeed, WPO with partial status subsumes WPO with total status and certain form of argument filtering. An argument filter π is said to be *non-collapsing* iff $\pi(f)$ is a list for every $f \in \mathcal{F}$.

Proposition 1. *Let π be a non-collapsing argument filter. For every \mathcal{F}^π -algebra \mathcal{A} and total status σ on \mathcal{F}^π , there exists an \mathcal{F} -algebra \mathcal{A}' and a partial status σ' on \mathcal{F} s.t.*

$$\langle \succsim_{\text{WPO}(\mathcal{A}, \sigma)}^\pi, \succ_{\text{WPO}(\mathcal{A}, \sigma)}^\pi \rangle = \langle \succsim_{\text{WPO}(\mathcal{A}', \sigma')}, \succ_{\text{WPO}(\mathcal{A}', \sigma')} \rangle$$

Proof. Let us define the interpretation of each $f \in \mathcal{F}_n$ in \mathcal{A}' by $f_{\mathcal{A}'}(\bar{x}_n) := f_{\mathcal{A}}(\bar{x}_{\pi(f)})$. Then obviously, $\pi(s) \succsim_{\mathcal{A}} \pi(t)$ iff $s \succsim_{\mathcal{A}'} t$. Moreover, we define $\sigma'(f)$ by $\pi(f) \star \sigma(f)$, where \star is a left-associative operator defined by

$$[a_1, \dots, a_n] \star [i_1, \dots, i_{n'}] := [a_{i_1}, \dots, a_{i_{n'}}]$$

Now we verify that $s \succsim_{\text{WPO}(\mathcal{A}, \sigma)}^\pi t$ implies $s \succsim_{\text{WPO}(\mathcal{A}', \sigma')}$ t by induction on $|s| + |t|$. If $s \in \mathcal{V}$, then $s = \pi(t) = t$ since π is non-collapsing, and hence $s \succsim_{\text{WPO}(\mathcal{A}', \sigma')} t$. Suppose $s = f(\bar{s}_n)$. We proceed by case analysis on the derivation of $\pi(s) \succsim_{\text{WPO}(\mathcal{A}, \sigma)} \pi(t)$.

1. Suppose $\pi(s) \succ_{\mathcal{A}} \pi(t)$. We obviously have $s \succ_{\mathcal{A}'} t$ and hence $s \succ_{\text{WPO}(\mathcal{A}', \sigma')} t$.
2. Suppose $\pi(s) \succsim_{\mathcal{A}} \pi(t)$. We obviously have $s \succsim_{\mathcal{A}'} t$.
 - (a) Suppose that $\pi(s) \succsim_{\text{WPO}(\mathcal{A}, \sigma)} \pi(t)$ is derived by case (2a). Then we have $s_i \succsim_{\text{WPO}(\mathcal{A}, \sigma)}^\pi t$ for some $i \in [1, \dots, n] \star \pi(f)$. By the induction hypothesis we have $s_i \succsim_{\text{WPO}(\mathcal{A}', \sigma')} t$, and since $i \in \sigma'(f)$, $s \succ_{\text{WPO}(\mathcal{A}', \sigma')} t$ by case (2a).
 - (b) Suppose that $\pi(s) \succsim_{\text{WPO}(\mathcal{A}, \sigma)} \pi(t)$ is derived by case (2b). Then we have $t = g(\bar{t}_m)$ and $s \succ_{\text{WPO}(\mathcal{A}, \sigma)}^\pi t_j$ for every $j \in [1, \dots, m] \star \pi(g)$. By the induction hypothesis we have $s \succ_{\text{WPO}(\mathcal{A}', \sigma')} t_j$, for all $j \in \sigma'(f)$. If furthermore $f \succ_{\mathcal{F}} g$, then immediately $s \succ_{\text{WPO}(\mathcal{A}', \sigma')} t$ by case (2b-i). If case (2b-ii) applies, then we obtain

$$[\bar{s}_n] \star \pi(f) \star \sigma(f) \succsim_{\text{WPO}(\mathcal{A}, \sigma)}^{\pi \text{ lex}} [\bar{t}_m] \star \pi(g) \star \sigma(g)$$

By the induction hypothesis and definition of σ' we obtain

$$[\bar{s}_{\sigma'(f)}] \succsim_{\text{WPO}(\mathcal{A}', \sigma')} [\bar{t}_{\sigma'(g)}] \quad \square$$

The advantage of partial status over argument filtering is due to the *weights* of ignored arguments. This is illustrated by the following example.

Example 15. [26] Consider a DP problem that induces the following constraints:

$$f^\sharp(s(x)) \succ f^\sharp(p(s(x))) \qquad p(s(x)) \lesssim x$$

In order to satisfy the first constraint by any simplification order, the argument of p must be filtered. However, the second constraint cannot be satisfied under such an argument filtering.

On the other hand, the DP problem can be shown finite using WPO with partial status s.t. $s_A(x) > x$, $f_A^\sharp(x) = p_A(x) = x$, $\sigma(f^\sharp) = \sigma(s) = [1]$, $\sigma(p) = []$, and $s >_{\mathcal{F}} p$. We have $f^\sharp(s(x)) \succ_{\text{WPO}} f^\sharp(p(s(x)))$ because of cases (2b-ii) and (2b-i), and $p(s(x)) \lesssim_{\text{WPO}} x$ because of case (1).

4.2. Refinements

As we will see in Section 7.2, the reduction pair processor induced by Definition 10 is already powerful in practice. However in theory, the WPO reduction pair is not a proper extension of the underlying interpretation, e.g., $x \lesssim_A g(x)$ if $g_A(x) = x$, but $x \lesssim_{\text{WPO}} g(x)$ cannot hold. Hence we refine the definition of \lesssim_{WPO} to properly subsume the underlying interpretation.

Note that in the above case, assuming $x \lesssim_{\text{WPO}} g(x)$ does not cause a problem if g is *least* (i.e., $f \gtrsim_{\mathcal{F}} g$ for every $f \in \mathcal{F}$) and $\sigma(g) = []$.

Proposition 2. *Let $g \in \mathcal{F}$ s.t. $f \gtrsim_{\mathcal{F}} g$ for every $f \in \mathcal{F}$ and $\sigma(g) = []$. Then $x \gtrsim_A t = g(\bar{t}_m)$ implies $s \lesssim_{\text{WPO}} t[x \mapsto s]$ for arbitrary non-variable terms $s = f(\bar{s}_n)$.*

Proof. By the definition of \gtrsim_A , we have $s \gtrsim_A t[x \mapsto s]$. Since g is least w.r.t. $\gtrsim_{\mathcal{F}}$, we have $f \gtrsim_{\mathcal{F}} g$. Moreover, since $\sigma(g) = []$, we have $[\bar{s}_{\sigma(f)}] \gtrsim_{\text{WPO}}^{\text{lex}} [\bar{t}_{\sigma(g)}] = []$. \square

Proposition 2 suggests a refined definition of $s \lesssim_{\text{WPO}} t$ by adding the following subcase in case (2) of Definition 10 (note that $s \gtrsim_A t$ is ensured in this case):

$$(2c) \ s \in \mathcal{V} \text{ and } t = g(\bar{t}_m) \text{ s.t. } \sigma(g) = [] \text{ and } g \text{ is least w.r.t. } \gtrsim_{\mathcal{F}}.$$

Similar refinements are proposed for KBO [8, 37] and for RPO [38], when t is a least constant. Our version is more general since t need not be a constant.

Example 16. [26] Consider a DP problem that induces the following constraints:

$$\begin{aligned} f^\sharp(s(x), y) &\succ f^\sharp(p(s(x)), p(y)) \\ f^\sharp(x, s(y)) &\lesssim f^\sharp(p(x), p(s(y))) \\ p(s(x)) &\lesssim x \end{aligned}$$

Let $\sigma(p) = []$, $\sigma(f^\sharp) = [1]$, $s_A(x) = x + 1$, $p_A(x) = x$, $f_A^\sharp(x, y) = x$ and p be least w.r.t. $\gtrsim_{\mathcal{F}}$. The first constraint is strictly oriented by case (2b-i). For the second constraint, it yields $x \lesssim_{\text{WPO}} p(x)$, for which case (2c) applies. Note that the argument of p cannot be filtered by an argument filter, because of the third constraint. Hence the refinements of [8, 37] or [38] do not work for this example.

Moreover, a further refinement is also possible for WPO, when the right-hand side is a variable. Note that $f(x) \succeq_{\mathcal{A}} x$ does not imply $f(x) \preceq_{\text{WPO}} x$ if $\sigma(f) = []$. Nonetheless, $f(x) \preceq_{\text{WPO}} x$ can be assumed if f is greatest, and moreover $f \sim_{\mathcal{F}} g$ implies $\sigma(g) = []$. The latter condition is crucial, since $g(x) \succ_{\text{WPO}} f(g(x))$ if $f \sim_{\mathcal{F}} g$ and $g = [1]$.

Proposition 3. *Suppose that \mathcal{A} is strictly simple w.r.t. σ , and $f \in \mathcal{F}$ s.t. either $f >_{\mathcal{F}} g$, or $f \sim_{\mathcal{F}} g$ and $\sigma(g) = []$ for every $g \in \mathcal{F}$. Then $s = f(\overline{s}_n) \succeq_{\mathcal{A}} y$ implies $s[y \mapsto t] \preceq_{\text{WPO}} t$ for arbitrary non-variable term $t = g(\overline{t}_m)$.*

Proof. By the definition of $\succeq_{\mathcal{A}}$, we have $s[y \mapsto t] \succeq_{\mathcal{A}} t$. Moreover by the strict simplicity, $s[y \mapsto t] \succeq_{\mathcal{A}} t >_{\mathcal{A}} t_j$ for all $j \in \sigma(g)$. Hence we get $s \succ_{\text{WPO}} t_j$. If $f >_{\mathcal{F}} g$, then $s[y \mapsto t] \succ_{\text{WPO}} t$ by case (2b-i). If $f \sim_{\mathcal{F}} g$ and $\sigma(g) = []$, then $s[y \mapsto t] \preceq_{\text{WPO}} t$ by case (2b-ii). \square

Provided \mathcal{A} is strictly simple w.r.t. σ , Proposition 3 suggests a refinement of $s \preceq_{\text{WPO}} t$ by adding the following subcase in case (2):

- (2d) $s = f(\overline{s}_n)$ and $t \in \mathcal{V}$ s.t. for every $g \in \mathcal{F}$, either $f >_{\mathcal{F}} g$ or $f \succeq_{\mathcal{F}} g$ and $\sigma(g) = []$.

Note that an arbitrary algebra is strictly simple w.r.t. the empty status. Hence WPO with the refinements can subsume even matrix interpretations. We obtain the following result:

Theorem 21. *Consider an instance of WPO that is induced by*

- *a well-founded algebra \mathcal{A} that is non-trivial, i.e., there exist $a, b \in A$ s.t. $a \not\preceq b$,*
- *the empty status function σ , and*
- *the quasi-precedence $\succeq_{\mathcal{F}}$ s.t. $f \succeq_{\mathcal{F}} g$ for arbitrary $f, g \in \mathcal{F}$.*

Then, $\langle \succeq_{\mathcal{A}}, >_{\mathcal{A}} \rangle = \langle \preceq_{\text{WPO}}, \succ_{\text{WPO}} \rangle$ after the refinements.

Proof. From the definition, it is obvious that $>_{\mathcal{A}} \subseteq \succ_{\text{WPO}}$ and $\preceq_{\text{WPO}} \subseteq \succeq_{\mathcal{A}}$. By the assumptions, cases (2b-i) and (2b-ii) of Definition 10 cannot apply for \succ_{WPO} . Hence we easily obtain $\succ_{\text{WPO}} \subseteq >_{\mathcal{A}}$.

Now suppose $s \succeq_{\mathcal{A}} t$ and let us show $s \preceq_{\text{WPO}} t$. The proof proceeds by case analysis on the structure of s and t .

- If $s, t \in \mathcal{V}$, then from non-triviality we have $s = t$, and hence $s \preceq_{\text{WPO}} t$.
- Suppose $s = f(\overline{s}_n)$ and $t = f(\overline{t}_m)$. Since $f >_{\mathcal{F}} g$ never holds, case (2b-i) of Definition 10 can be ignored. Moreover, since $f \succeq_{\mathcal{F}} g$ and $[] \preceq_{\text{WPO}}^{\text{lex}} []$, $s \succeq_{\mathcal{A}} t$ implies $s \preceq_{\text{WPO}} t$.
- If either s or t is a variable, then refinement (2c) or (2d) is satisfied. Hence $s \preceq_{\text{WPO}} t$. \square

In the appendix, we prove soundness of WPO with the refinements (2c) and (2d).

Theorem 22 (Soundness). *If \mathcal{A} is weakly monotone and weakly simple w.r.t. σ , then $\langle \preceq_{\text{WPO}}, \succ_{\text{WPO}} \rangle$ forms a reduction pair.*

4.3. Comparison with Other Reduction Pairs

In this section, we investigate some relationships between instances of WPO and existing reduction pairs.

In Definition 10, it is obvious that the induced \succ_{WPO} is identical to that induced by Definition 5, if we choose a *total* status σ . Hence Theorems 16, 18 and 19 imply that WPO subsumes KBO, TKBO and LPO resp. also as a reduction pair.

On the other hand, Theorem 17 does not imply that $\text{WPO}(\mathcal{P}ol)$ subsumes POLO as a reduction pair, since the “weak-part” $\geq_{\mathcal{P}ol}$ is not considered in the theorem. Nonetheless, after the refinements in Section 4.2, we obtain the following result from Theorem 21:

Corollary 5. *WPO($\mathcal{P}ol$) with the refinements (2c) and (2d) subsumes POLO.* \square

It is now easy to obtain the following result:

Corollary 6. *WPO($\mathcal{M}Pol$) with the refinements (2c) and (2d) subsumes POLO, KBO, TKBO and LPO.* \square

Moreover, WPO also subsumes the *matrix interpretation method* [21], when weights are computed by a matrix interpretation \mathcal{Mat} . Note that a matrix interpretation is not always weakly simple. Hence as a *reduction order*, Definition 5 cannot be applied for \mathcal{Mat} in general. The situation is relaxed for reduction pairs, and from Theorem 21 we obtain the following:

Corollary 7. *WPO(\mathcal{Mat}) with the refinements (2c) and (2d) subsumes the reduction pair induced by the matrix interpretation \mathcal{Mat} .* \square

Finally, we compare $\text{WPO}(\mathcal{M}Pol)$ and $RPOLO$ of Bofill *et al.* [22], another approach of unifying LPO and POLO. It turns out that $RPOLO$ is incomparable with $\text{WPO}(\mathcal{M}Pol)$. First we verify that $\text{WPO}(\mathcal{M}Pol)$ is not subsumed by $RPOLO$; more precisely, $RPOLO$ does not subsume KBO.

Example 17. Let us show that the constraint $f(g(x)) \succ g(f(f(x)))$ cannot be satisfied by $RPOLO$.⁵ Note that this constraint is satisfied by KBO with $w(f) = 0$ and $f >_{\mathcal{F}} g$.

- Suppose $f \in \mathcal{F}_{\text{POLO}}$. Since this constraint cannot be satisfied by POLO, g must be in $\mathcal{F}_{\text{RPOLO}}$. Hence we need $f_{\mathcal{P}ol}(v_{g(x)}) >_{C(f(g(x)))} v_{g(f(f(x)))}$. This requires either

- $f_{\mathcal{P}ol}(x) = x$ and $g(x) \succ_{\text{RPOLO}} g(f(f(x)))$, or
- $f_{\mathcal{P}ol}(x) > x$ and $g(x) \succeq_{\text{RPOLO}} g(f(f(x)))$.

In either case, we obtain $g(x) \succ_{\text{RPOLO}} g(x)$, which is a contradiction.

⁵ To simplify the discussion, we do not consider the possibility for *argument filterings* or *usable rules* [18] in the following examples. It is easy to exclude these techniques; for example, by adding the constraint $g(x) \lesssim x$ we can enforce the argument of g not to be filtered.

- Suppose $f \in \mathcal{F}_{\text{RPO}}$. Since this constraint cannot be satisfied by RPO, g must be in $\mathcal{F}_{\text{POLO}}$. Hence we need

- $g(x) \succeq_{\text{RPOLO}} g(f(f(x)))$, or
- $f(g(x)) \succ_{\text{RPOLO}} f(f(x))$.

The first case contradicts $f(x) \succ_{\text{RPOLO}} x$. The second case contradicts the fact that $f(x) \succ_{\text{RPOLO}} g(x)$.

On the other hand, RPOLO is also not subsumed by $\text{WPO}(\mathcal{MPol})$, as the following example illustrates.

Example 18. Consider a DP problem that induces the following constraints:

$$f^\sharp(i(x, i(y, g(z)))) \succ f^\sharp(i(y, i(z, x))) \quad (1)$$

$$f(g(h(x))) \preceq g(f(h(g(x)))) \quad (2)$$

$$i(y, i(z, x)) \preceq i(x, i(y, z)) \quad (3)$$

where constraint (2) is from [31, Proposition 10].

- First, let us show that the set of constraints cannot be satisfied by $\text{WPO}(\mathcal{MPol})$. Since f, g and h are unary, we only consider $ws(f) = ws(g) = ws(h) = \text{pol}$. It is easy to adjust [31, Proposition 10] to show that $g_{\text{Pol}}(x) = x$ whenever (2) is satisfied. Together with (3), we obtain

$$i(y, i(z, x)) \geq_{\mathcal{MPol}} i(x, i(y, z)) =_{\mathcal{MPol}} i(x, i(y, g(z)))$$

Hence case (1) of Definition 10 cannot be applied for constraint (1). Moreover, by any choice of $\sigma(i)$, case (2b–ii) cannot apply, either.

- Second, let us show that the set of constraints can be satisfied by RPOLO. Consider $f, g, h \in \mathcal{F}_{\text{RPO}}$, $i, f^\sharp \in \mathcal{F}_{\text{POLO}}$, $f >_{\mathcal{F}} g >_{\mathcal{F}} h$, $i_{\text{Pol}}(x, y) = x + y$ and $f^\sharp_{\text{Pol}}(x) = x$. Then constraint (2) is strictly oriented and (3) is weakly oriented. Since $g(z) \succ_{\text{RPOLO}} z$ and $v_{g(z)} > z$ implies $x + y + v_{g(z)} > y + z + x$, constraint (1) is also satisfied. \square

5. SMT Encodings

In the preceding sections, we have concentrated on theoretical aspects. In this section, we consider how to implement the instances of WPO using SMT solvers. We extend the corresponding approach for KBO [9] to WPO. In particular, $\text{WPO}(\text{Sum})$, $\text{WPO}(\text{Max})$ and $\text{WPO}(\text{MSum})$ are reduced to SMT problems of linear arithmetic, and as a consequence, decidability is ensured for orientability problems of these orders.

An *expression* e is built from (non-negative integer) variables, constants and the binary symbols \cdot and $+$ denoting multiplication and addition, resp. A *formula* is built from atoms of the form $e_1 > e_2$ and $e_1 \geq e_2$, negation \neg , and the binary symbols \wedge , \vee and \Rightarrow denoting conjunction, disjunction and implication, resp. The precedence of these symbols is in the order we listed above.

The main interest of the SMT encoding approach is to employ SMT solvers for finding a concrete algebra that proves finiteness of a given DP problem (or termination of a given TRS). Hence we assume that algebras are parameterized by a set of expression variables.

Definition 11. An algebra \mathcal{A} *parameterized* by a set V of variables is a mapping that induces a concrete algebra \mathcal{A}^α from an assignment α whose domain contains V . An *encoding* of the relation $\succsim_{\mathcal{A}}$ is a function that assigns for two terms s and t a formula $\llbracket s \succsim_{\mathcal{A}} t \rrbracket$ over variables from V s.t. $\alpha \models \llbracket s \succsim_{\mathcal{A}} t \rrbracket$ iff $s \succsim_{\mathcal{A}^\alpha} t$.

In the encodings presented in the rest of this section, we consider \mathcal{A} to be parameterized by at most the following variables:

- integer variables w_f and w_0 denoting $w(f)$ and w_0 , resp., and
- integer variables $sc_{f,i}$ and $sp_{f,i}$ denoting $sc(f, i)$ and $sp(f, i)$, resp.

5.1. The Common Structure

To optimize the presentation, we present an encoding of the common structure of WPO independent from the shape of \mathcal{A} . Hence, we assume encodings for $>_{\mathcal{A}}$ and $\succsim_{\mathcal{A}}$ are given.

Following [9], first we represent a quasi-precedence $\succsim_{\mathcal{F}}$ by integer variables p_f . For an assignment α , we define the quasi-precedence $\succsim_{\mathcal{F}}^\alpha$ as follows: $f \succsim_{\mathcal{F}}^\alpha g$ iff $\alpha \models p_f \geq p_g$.

Next we consider representing a partial status by imitating the encoding of a *filtered permutation* proposed in [6]. We introduce the boolean variables $st_{f,i}$ and $st_{f,i,j}$, so that an assignment α induces a status σ^α as follows: $\alpha \models st_{f,i}$ iff $i \in \sigma^\alpha(f)$, and $\alpha \models st_{f,i,j}$ iff i is the j -th element in $\sigma^\alpha(f)$. In order for σ^α to be well-defined, every $i \in \sigma^\alpha(f)$ must occur exactly once in $\sigma^\alpha(f)$ and any $i \notin \sigma^\alpha(f)$ must not occur in $\sigma^\alpha(f)$. These conditions are represented by the following formula:

$$ST := \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=1}^n \left(\left(st_{f,i} \Rightarrow \sum_{j=1}^n st_{f,i,j} = 1 \right) \wedge \left(\neg st_{f,i} \Rightarrow \sum_{j=1}^n st_{f,i,j} = 0 \right) \right)$$

It is easy to verify that σ^α is well-defined if $\alpha \models ST$. In contrast to the previous works [9, 6], we moreover need the following formula to ensure weak simplicity of \mathcal{A} w.r.t. σ :

$$SIMP := \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=1}^n \left(st_{f,i} \Rightarrow \llbracket f(\overline{x}_n) \succsim_{\mathcal{A}} x_i \rrbracket \right)$$

Note that in the formula $SIMP$, the condition $f(\overline{x}_n) \succsim_{\mathcal{A}} x_i$ can often be encoded in a more efficient way. For linear \mathcal{Pol} , for example, this condition is equivalent to $sc(f, i) \geq 1$.

Lemma 3. $\alpha \models ST \wedge SIMP$ iff \mathcal{A}^α is weakly simple w.r.t. a partial status σ^α . \square

Now we present the encodings for WPO.

Definition 12. The encodings of \succ_{WPO} and \succsim_{WPO} are defined as follows:

$$\llbracket s \succ_{WPO} t \rrbracket := \llbracket s >_{\mathcal{A}} t \rrbracket \vee (\llbracket s \succsim_{\mathcal{A}} t \rrbracket \wedge s \succsim_1 t)$$

where the formula $s \lesssim_1 t$ is defined as follows:

$$\begin{aligned} x \lesssim_1 t &:= \begin{cases} \text{TRUE} & \text{if } x = t \\ \text{FALSE} & \text{otherwise} \end{cases} \\ x \succ_1 t &:= \text{FALSE} \\ f(\bar{s}_n) \lesssim_1 t &:= \bigvee_{i=1}^n \left(\text{st}_{f,i} \wedge \llbracket s_i \lesssim_{\text{WPO}} t \rrbracket \right) \vee f(\bar{s}_n) \lesssim_2 t \end{aligned}$$

The formula $f(\bar{s}_n) \lesssim_2 t$ is defined as follows:

$$\begin{aligned} f(\bar{s}_n) \lesssim_2 y &:= \text{FALSE} \\ f(\bar{s}_n) \lesssim_2 g(\bar{t}_m) &:= \bigwedge_{j=1}^m \left(\text{st}_{g,j} \Rightarrow \llbracket s \succ_{\text{WPO}} t_j \rrbracket \right) \wedge \\ &\quad \left(\mathbf{p}_f > \mathbf{p}_g \vee \mathbf{p}_f = \mathbf{p}_g \wedge \llbracket [\bar{s}_{\sigma(f)}] \lesssim_{\text{WPO}}^{\text{lex}} [\bar{t}_{\sigma(g)}] \rrbracket \right) \end{aligned}$$

Here, $s \lesssim_1 t$ indicates that $s \lesssim_{\text{WPO}} t$ is derived by case (2a) or (2b), and $s \lesssim_2 t$ indicates that $s \lesssim_{\text{WPO}} t$ is derived by case (2b-i) or (2b-ii). We do not present the encoding for the lexicographic extension w.r.t. permutation, which can be found in [6].

For an assignment α , we write $\lesssim_{\text{WPO}}^\alpha$ to denote the instance of WPO corresponding to α ; i.e., $\lesssim_{\text{WPO}}^\alpha$ is induced by the algebra \mathcal{A}^α , the quasi-precedence $\succ_{\mathcal{F}}^\alpha$ and the partial status σ^α .

Lemma 4. *For any assignment α s.t. $\alpha \models \text{ST} \wedge \text{SIMP}$, $\alpha \models \llbracket s \lesssim_{\text{WPO}} t \rrbracket$ iff $s \lesssim_{\text{WPO}}^\alpha t$.* \square

Theorem 23. *If the following formula is satisfiable:*

$$\text{ST} \wedge \text{SIMP} \wedge \bigwedge_{l \rightarrow r \in \mathcal{R} \cup \mathcal{P}} \llbracket l \lesssim_{\text{WPO}} r \rrbracket \wedge \bigvee_{l \rightarrow r \in \mathcal{P}'} \llbracket l \succ_{\text{WPO}} r \rrbracket \quad (4)$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound.

Proof. Let α be the assignment that satisfies (4). By Lemma 4, we obtain $l \lesssim_{\text{WPO}}^\alpha r$ for all $l \rightarrow r \in \mathcal{R} \cup \mathcal{P}$ and $l \succ_{\text{WPO}}^\alpha r$ for all $l \rightarrow r \in \mathcal{P}'$. Moreover by Theorem 22, $\langle \lesssim_{\text{WPO}}^\alpha, \succ_{\text{WPO}}^\alpha \rangle$ forms a reduction pair. Hence Theorem 9 concludes the soundness of this DP processor. \square

In the following sections, we give encodings depending on the choice of \mathcal{A} for each instance of WPO.

5.2. Encoding $\text{WPO}(\text{Pol})$ and $\text{WPO}(\text{Sum})$

First we present an encoding of a linear polynomial interpretation Pol . The encodings for Sum is obtained by fixing all coefficients to 1. The weight of a term s and the variable coefficient of x in s are encoded as follows:

$$\begin{aligned} \mathbf{w}(s) &:= \begin{cases} \mathbf{w}_0 & \text{if } s \in \mathcal{V} \\ \mathbf{w}_f + \sum_{i=1}^n \mathbf{sc}_{f,i} \cdot \mathbf{w}(s_i) & \text{if } s = f(\overline{s}_n) \end{cases} \\ \mathbf{vc}(x, s) &:= \begin{cases} 1 & \text{if } x = s \\ 0 & \text{if } x \neq s \in \mathcal{V} \\ \sum_{i=1}^n \mathbf{sc}_{f,i} \cdot \mathbf{vc}(x, s_i) & \text{if } s = f(\overline{s}_n) \end{cases} \end{aligned}$$

We have to ensure \mathbf{w}_0 to be the lower bound of weights of terms. To ensure $\mathbf{w}(f(\overline{s}_n)) \geq \mathbf{w}_0$ for every term $f(\overline{s}_n)$, we need either $\mathbf{w}_f \geq \mathbf{w}_0$ or one of the arguments to have a positive coefficient (note that the weight of this argument is at least \mathbf{w}_0). This is represented by the following constraint:

$$\mathbf{WMIN} := \bigwedge_{f \in \mathcal{F}_n} \left(\mathbf{w}_f \geq \mathbf{w}_0 \vee \bigvee_{i=1}^n \mathbf{sc}_{f,i} \geq 1 \right)$$

Now the relations $>_{\mathcal{P}ol}$ and $\geq_{\mathcal{P}ol}$ are encoded as follows:

$$\llbracket s \geq_{\mathcal{P}ol} t \rrbracket := \mathbf{w}(s) \geq \mathbf{w}(t) \wedge \bigwedge_{x \in \text{Var}(t)} \mathbf{vc}(x, s) \geq \mathbf{vc}(x, t)$$

Corollary 8. *If the following formula is satisfiable:*

$$\mathbf{ST} \wedge \mathbf{SIMP} \wedge \mathbf{WMIN} \wedge \bigwedge_{l \rightarrow r \in \mathcal{R} \cup \mathcal{P}} \llbracket l \prec_{\text{WPO}(\mathcal{P}ol)} r \rrbracket \wedge \bigvee_{l \rightarrow r \in \mathcal{P}'} \llbracket l \succ_{\text{WPO}(\mathcal{P}ol)} r \rrbracket$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound. \square

5.3. Encoding $\text{WPO}(\text{Max})$

In this section, we consider encoding $\text{WPO}(\text{Max})$. Unfortunately, we are aware of no SMT solver which supports a built-in max operator. Hence we consider encoding the constraint $s >_{\text{Max}} t$ into both quantified and quantifier-free formulas.

First, we present an encoding to a quantified formula. A straightforward encoding would involve

$$\mathbf{w}(s) := \begin{cases} s & \text{if } s \in \mathcal{V} \\ v & \text{if } s = f(\overline{s}_n) \end{cases}$$

where v is a fresh integer variable representing $\max\{\mathbf{w}_f, \mathbf{w}(s_1), \dots, \mathbf{w}(s_n)\}$ with the following constraint ϕ added into the context:

$$\phi := v \geq \mathbf{w}_f \wedge \bigwedge_{i=1}^n v \geq \mathbf{w}(s_i) \wedge \left(v = \mathbf{w}_f \vee \bigvee_{i=1}^n v = \mathbf{w}(s_i) \right)$$

Then the constraint $s \geq_{\text{Max}} t$ can be encoded as follows:

$$\llbracket s \geq_{\text{Max}} t \rrbracket := \forall \overline{x}_k, \overline{v}_m. \phi_1 \wedge \dots \wedge \phi_m \Rightarrow \mathbf{w}(s) \geq \mathbf{w}(t)$$

where $\{x_1, \dots, x_k\} = \text{Var}(s) \cup \text{Var}(t)$ and each $\langle \phi_j, v_j \rangle$ is the pair of the constraint and the fresh variable introduced during the encoding.

Although quantified linear integer arithmetic is known to be decidable, the SMT solvers we have tested could not solve the problems generated by the above straightforward encoding efficiently, if at all. Fuhs *et al.* [16] propose a sound elimination of quantifiers by introducing new template polynomials. Here we propose another encoding to quantifier-free formulas that does not introduce extra polynomials and is sound and complete for linear polynomials with max.

Definition 13. A *generalized weight* [39] is a pair $\langle n, N \rangle$ where $n \in \mathbb{N}$ and N is a finite multiset⁶ over \mathcal{V} . We define the following operations:

$$\begin{aligned}\langle n, N \rangle + \langle m, M \rangle &:= \langle n + m, N \uplus M \rangle \\ n \cdot \langle m, M \rangle &:= \langle n \cdot m, n \cdot M \rangle\end{aligned}$$

where $n \cdot M$ denotes the multiset that maps x to $n \cdot M(x)$ for every $x \in \mathcal{V}$. We encode a generalized weight as a pair of an expression and a mapping N from \mathcal{V} to expressions s.t. the *domain* $\text{Dom}(N) := \{x \mid N(x) \neq 0\}$ of N is finite. Notations for generalized weights are naturally extended for encoded ones. The relation \supseteq on multisets is encoded as follows:

$$N \supseteq M := \bigwedge_{x \in \text{Dom}(M)} N(x) \geq M(x)$$

A generalized weight $\langle n, N \rangle$ represents the expression $n + \sum_{x \in N} x$. Now we consider removing max.

Definition 14. The *expanded weight* $\overline{\mathbf{w}}(s)$ of a term s induced by a weight function $\langle w, w_0 \rangle$ and a subterm penalty function sp is a set of generalized weights, which is defined as follows:

$$\overline{\mathbf{w}}(s) := \begin{cases} \{\langle w_0, \{s\} \rangle\} & \text{if } s \in \mathcal{V} \\ \{\langle w_f, \emptyset \rangle\} \cup \{\langle \mathbf{sp}_{f,i} + p \mid p \in \overline{\mathbf{w}}(s_i), 1 \leq i \leq n \rangle\} & \text{if } s = f(\overline{s}_n) \end{cases}$$

The expanded weight $\overline{\mathbf{w}}(s) = \{\overline{p}_n\}$ represents the expression $\max\{\overline{e}_n\}$, where each generalized weight p_i represents the expression e_i . Using expanded weights, we can encode $>_{\mathcal{M}ax}$ and $\geq_{\mathcal{M}ax}$ in a way similar to the *max set ordering* presented in [40]:

$$\llbracket s \geq_{\mathcal{M}ax} t \rrbracket := \bigwedge_{\langle m, M \rangle \in \overline{\mathbf{w}}(t)} \bigvee_{\langle n, N \rangle \in \overline{\mathbf{w}}(s)} (n \geq m \wedge N \supseteq M)$$

Using the quantified or quantifier-free encodings, we obtain the following corollary of Theorem 22:

Corollary 9. *If the following formula is satisfiable:*

$$\text{ST} \wedge \bigwedge_{l \rightarrow r \in \mathcal{R} \cup \mathcal{P}} \llbracket l \lesssim_{\text{WPO}(\mathcal{M}ax)} r \rrbracket \wedge \bigvee_{l \rightarrow r \in \mathcal{P}'} \llbracket l \succ_{\text{WPO}(\mathcal{M}ax)} r \rrbracket$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound. □

⁶In the encoding for $\mathcal{M}ax$, N need not contain more than one variable. This generality is reserved for the encoding of $\mathcal{M}Pol$.

5.4. Encoding $WPO(\mathcal{MPol})$ and $WPO(\mathcal{MSum})$

In this section, we consider encoding linear polynomials with max into SMT formulas. First we extend Definition 14 for weight statuses.

Definition 15. For a weight status ws , the *expanded weight* $\overline{w}^{ws}(s)$ of a term s is the set of generalized weight, which is recursively defined as follows:

$$\overline{w}^{ws}(s) := \begin{cases} \{(w_0, \{s\})\} & \text{if } s \in \mathcal{V} \\ S & \text{if } s = f(\overline{s}_n), \text{ } ws(f) = \text{pol} \\ T & \text{if } s = f(\overline{s}_n), \text{ } ws(f) = \text{max} \end{cases}$$

where

$$S = \left\{ w_f + \sum_{i=1}^n \text{sc}_{f,i} \cdot p_i \mid p_1 \in \overline{w}^{ws}(s_1), \dots, p_n \in \overline{w}^{ws}(s_n) \right\}$$

$$T = \{w_f\} \cup \{\text{sp}_{f,i} + \text{sc}_{f,i} \cdot p \mid p \in \overline{w}^{ws}(s_i), i \in \{1, \dots, n\}\}$$

Now the encoding of $>_{\mathcal{MPol}}$ and $\geq_{\mathcal{MPol}}$ are given as follows:

$$\llbracket s \geq_{\mathcal{MPol}} t \rrbracket := \bigwedge_{\langle m, M \rangle \in \overline{w}^{ws}(t)} \bigvee_{\langle n, N \rangle \in \overline{w}^{ws}(s)} (n \geq m \wedge N \supseteq M)$$

Corollary 10. *If the following formula is satisfiable:*

$$\text{ST} \wedge \text{SIMP} \wedge \text{WMIN} \wedge \bigwedge_{l \rightarrow r \in \mathcal{R} \cup \mathcal{P}} \llbracket l \succ_{WPO(\mathcal{MPol})} r \rrbracket \wedge \bigvee_{l \rightarrow r \in \mathcal{P}'} \llbracket l \succ_{WPO(\mathcal{MPol})} r \rrbracket$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound. \square

5.5. Encoding $WPO(\mathcal{Mat})$

We omit presenting an encoding of the matrix interpretation method, which can be found in [21]. In order to use a matrix interpretation in WPO, however, small care is needed; one has to ensure weak simplicity of \mathcal{Mat} w.r.t. σ . This can be done as follows:

Lemma 5. *If $SC(f, i)^{j,j} \geq 1$ for all $f \in \mathcal{F}_n$, $i \in \sigma(f)$ and $j \in \{1, \dots, d\}$, then \mathcal{Mat} is weakly simple w.r.t. σ .* \square

5.6. Encoding for Reduction Orders

In case one wants an encoding for the reduction order $>_{WPO}$ defined in Definition 5, then the status σ must be total. This can be ensured by enforcing $i \in \sigma(f)$ for all $i \in \{1, \dots, n\}$ and $f \in \mathcal{F}$, which is represented by the following formula:

$$\text{TOTAL} := \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=1}^n \text{st}_{f,i}$$

or equivalently by replacing all $\text{st}_{f,i}$ by TRUE . Note that $\text{TOTAL} \wedge \text{SIMP}$ enforces all the subterm coefficients to be greater than or equal to 1.

Theorem 24. *If the following formula is satisfiable:*

$$\text{TOTAL} \wedge \text{ST} \wedge \text{SIMP} \wedge \text{WMIN} \wedge \bigwedge_{l \rightarrow r \in \mathcal{R}} \llbracket l \succ_{WPO(\mathcal{MPol})} r \rrbracket$$

then \mathcal{R} is orientable by $WPO(\mathcal{MPol})$. \square

6. Optimizations

In our implementation, some optimizations are performed during the encoding. For example, formulas like $\text{FALSE} \wedge \phi$ are reduced in advance to avoid generating meaningless formulas, and temporary variables are inserted to avoid multiple occurrences of an expression or a formula. Moreover, we apply several optimizations that we discuss below.

6.1. Fixing w_0

We can simplify the encoded formulas by fixing w_0 . For KBO, Winkler *et al.* [13] show that w_0 can be fixed to an arbitrary $k > 0$ (e.g., 1) without losing any power. By adapting the proof of [13, Lemma 3], it can be shown that w_0 can be fixed to 0 for $\text{WPO}(\text{Sum})$. On the contrary to KBO, however, fixing $w_0 > 0$ will affect the power, since the transformation of [13] may assign negative weights to some symbols when applied to the case $w_0 = 0$.

6.2. Fixing Weight Status

For POLO and WPO using algebras \mathcal{MSum} and \mathcal{MPol} , it may not be practical to consider all possible weight statuses. Hence, we introduce a heuristic for fixing ws . In case of $\text{WPO}(\mathcal{MSum})$, ws should at least satisfy the following condition for all $l \rightarrow r \in \mathcal{R} \cup \mathcal{P}$:

$$\bigwedge_{\langle m, M \rangle \in \overline{w}^{ws}(r)} \bigvee_{\langle n, N \rangle \in \overline{w}^{ws}(l)} N \supseteq M$$

since otherwise the formula $\bigwedge_{l \rightarrow r \in \mathcal{R}} \llbracket l \lesssim_{\text{WPO}(\mathcal{MSum})} r \rrbracket$ is trivially unsatisfiable. Hence in our implementation, we require that \mathcal{MSum} and \mathcal{MPol} are induced by the weight status which minimizes the number of f with $ws(f) = \text{max}$, while satisfying the above condition.

6.3. Reducing Recursive Checks

Encoding KBO as a reduction order [9] is notably efficient, because KBO does not have recursive checks like LPO or WPO. For WPO, we can reduce formulas for recursive checks by restricting $w_0 > 0$, since under this restriction, $f(\overline{s}_n) >_{\mathcal{MPol}} s_i$ holds whenever $n \geq 2$ and $ws(f) = \text{pol}$. Hence if $n \geq 2$ and $ws(f) = \text{pol}$, we reduce the formula $f(\overline{s}_n) \lesssim_1 t$ to $f(\overline{s}_n) \lesssim_2 t$. Analogously if $m \geq 2$ and $ws(g) = \text{pol}$, we reduce the formula $f(\overline{s}_n) \lesssim_2 g(\overline{t}_m)$ to the following:

$$\mathbf{p}_f > \mathbf{p}_g \vee \mathbf{p}_f = \mathbf{p}_g \wedge \llbracket [\overline{s}_{\sigma(f)}] \lesssim_{\text{WPO}(\mathcal{A}, \sigma)}^{\text{lex}} [\overline{t}_{\sigma(g)}] \rrbracket$$

without generating formulas for recursive checks corresponding to cases (2a) and (2b). Note however that this simplification does not apply when encoding reduction pairs using argument filtering, as we will see in Section 7.2.

7. Experiments

In this section we examine the performance of WPO both as a reduction order and in the DP framework. We implemented a simple form of the DP framework as the *Nagoya Termination Tool* (**NaTT**) and incorporated WPO as a DP processor [48]. The encodings presented in Section 5 and optimizations

Table 1: Results for Reduction Orders

order	algebra	non-dup. TRSs			dup. TRSs		
		yes	T.O.	time	yes	T.O.	time
POLO	$\mathcal{S}um$	41	0	4.45	–	–	–
POLO	$\mathcal{MS}um$	60	0	4.46	19	0	–
LPO		90	0	31.64	90	0	35.39
KBO		115	0	6.20	–	–	–
WPO	$\mathcal{S}um^+$	126	0	6.70	–	–	–
WPO	$\mathcal{S}um$	135	0	43.16	–	–	–
WPO	$\mathcal{M}ax$	109	0	53.77	125	0	49.49
WPO	$\mathcal{MS}um$	135	0	42.72	138	0	66.15
POLO	$\mathcal{P}ol$	104	3	203.37	21	10	1065.34
POLO	$\mathcal{MP}ol$	104	3	203.07	39	8	608.76
TKBO		132	3	226.33	27	12	1414.62
WPO	$\mathcal{P}ol$	149	3	280.92	29	12	1495.55
WPO	$\mathcal{MP}ol$	149	3	280.86	138	9	1008.67
POLO+KBO+LPO		130	0	35.35	92	0	51.35

presented in Section 6 are implemented. In the encodings of $\mathcal{P}ol$ and $\mathcal{MP}ol$, we choose 3 as upper bound of weights and coefficients, in order to achieve a practical runtime. For comparison, KBO, TKBO, LPO, polynomial interpretations with or without max and matrix interpretations are implemented in the same manner. For the DP framework, we implemented the estimation of *dependency graphs* in [41], and *strongly connected components* are sequentially processed in order of size where smaller ones come first. Moreover, *usable rules* w.r.t. argument filters are also implemented by following the encoding proposed in [42].

The test set of termination problems are the 1463 TRSs from the TRS Standard category of TPDB 8.0.6 [27]. The experiments are run on a server equipped with two quad-core Intel Xeon W5590 processors running at a clock rate of 3.33GHz and 48GB of main memory, though only one thread of SMT solver runs at once. As the SMT solver, we choose z3 4.3.1.⁷ Timeout is set to 60s, as in the *Termination Competition* [43]. Details of the experimental results are available at <http://www.trs.cm.is.nagoya-u.ac.jp/papers/SCP2014/>.

7.1. Results for Reduction Orders

First we evaluated WPO as a reduction order by directly testing orientability for input TRSs. The results are listed in Table 1. Since KBO, POLO($\mathcal{S}um$), WPO($\mathcal{S}um$) are only applicable for non-duplicating TRSs, the test set is split into non-duplicating ones (consisting of 439 TRSs) and duplicating ones (consisting of 1024 TRSs). In the table, the ‘yes’ column indicates the number of successful termination proofs, ‘T.O.’ indicates the number of timeouts, and ‘time’ indicates the total time. To emphasize the benefit of WPO, we also compare it with arbitrary lexicographic compositions of POLO, KBO, and LPO (‘POLO+KBO+LPO’ row).

⁷<http://z3.codeplex.com/>

Table 2: Results for Reduction Pairs

order	algebra	total status			partial status		
		yes	T.O.	time	yes	T.O.	time
POLO	$\mathcal{S}um$	512	0	150.99	–	–	–
POLO	$\mathcal{MS}um$	522	0	300.65	–	–	–
LPO		502	0	435.62	–	–	–
KBO		497	3	1001.55	520	4	1238.67
WPO	$\mathcal{S}um$	514	3	907.27	560	5	1244.04
WPO	$\mathcal{M}ax$	548	7	1269.48	637	13	1846.06
WPO	$\mathcal{MS}um$	578	5	1261.63	675	12	1827.01
POLO	$\mathcal{P}ol$	544	19	1958.44	–	–	–
POLO	$\mathcal{MP}ol$	540	18	1889.86	–	–	–
POLO	$\mathcal{M}at$	645	480	32367.26	–	–	–
TKBO		516	187	15665.26	539	178	15799.28
WPO	$\mathcal{P}ol$	527	172	14535.24	579	153	13579.95
WPO	$\mathcal{MP}ol$	560	88	7678.43	672	94	9269.36
WPO	$\mathcal{M}at$	–	–	–	538	640	42067.45
THOR		418	261	18550.62	–	–	–

We point out that $WPO(\mathcal{MS}um)$ is a balanced choice; it is significantly stronger than existing orders, while the runtime is much better than involving non-linear SMT solving (last 5 columns). Note that we directly solve non-linear problems using `z3`; it may be possible to improve efficiency by e.g., a SAT encoding like [17]. In that case, we expect $WPO(\mathcal{MP}ol)$ to become a practical choice. If efficiency is the main concern, then $WPO(\mathcal{S}um^+)$, a variant of $WPO(\mathcal{S}um)$ with $w_0 > 0$, is a reasonable substitute for KBO. This efficiency is due to the reduction of recursive checks proposed in Section 6.3.

7.2. Results for Reduction Pairs

Second, we evaluated WPO as a reduction pair. Table 2 compares the power of the reduction pair processors. In ‘total status’ column, we apply standard total statuses and argument filtering to obtain a reduction pair from a reduction order, as in [28]. Because of argument filtering, the existence of duplicating rules is not an issue in this setting. For POLO, statuses and argument filtering are ignored (the latter is considered as 0-coefficient). The power of WPO is still measurable here. On the contrary to the reduction order case, $WPO(\mathcal{S}um)$ outperforms KBO both in power and efficiency. This is because KBO needs formulas for recursive comparison that resembles WPO, when argument filters are considered. Moreover, encodings of weights are more complex in KBO, since w_0 cannot be fixed to 0 as discussed in Section 6.1. Finally, KBO needs extra constraints that correspond to admissibility.

In ‘partial status’ column, we moreover admit partial statuses of Definition 10. We also apply partial status for KBO as in [26] but not for LPO, since LPO does not benefit from partial statuses because weights are not considered. The power of WPO is much more significant in this setting, and $WPO(\mathcal{MS}um)$ is about 30% stronger than any other existing techniques. Though the efficiency is sacrificed for partial statuses, this is not a severe problem in the DP framework, as we will see in the next section. On the other hand, our implementation

Table 3: Results for Combination

tools	yes	no	maybe	T.O.	time
NaTT	848	173	429	13	1865.50
NaTT w/o WPO	810	173	467	13	2023.18
AProVE	1020	271	0	173	15123.48
$\mathsf{T}_1\mathsf{T}_2$	788	193	417	65	13784.43

of the instances of WPO that require non-linear SMT solving are extremely time-consuming. Especially, $\text{WPO}(\mathcal{M}at)$ loses 107 problems by timeout compared to the standard matrix interpretation method. We conjecture that the situation can be improved by SAT encoding or by using other non-linear SMT solvers such as [44, 45].

In order to estimate the power of RPOLO, we also ran an experiment with THOR,⁸ the only termination prover having RPOLO implemented, as far as we know. Note however that it might be unfair to compare the results directly; THOR is specialized to higher-order case and is based on MSPO, while our implementation is based on the DP framework.

7.3. Combining DP Processors

In modern termination provers, DP processors are combined in the DP framework and weak but efficient ones are applied first. The default strategy of NaTT sequentially applies the *rule removal processor* [34], the (*generalized*) *uncurrying* [46, 47], reduction pair processors including standard POLO, LPO, POLO with max [16] and $\text{WPO}(\mathcal{M}Sum)$ with partial status, and then a simple variant of the matrix interpretation method [21]. When all reduction pair processors fail, a naive loop detection is performed to conclude nontermination. In Table 3, we compare the following settings: ‘NaTT’ (the default strategy described above), ‘NaTT w/o WPO’ (WPO is replaced by KBO),⁹ AProVE 2014, and $\mathsf{T}_1\mathsf{T}_2$ 1.15. The ‘no’ column indicates the number of successful nontermination proofs. In this setting, NaTT discovered termination proofs for 36 of 159 problems whose termination could not be proved by any other tools participated in the full-run of the Termination Competition 2013 [43]. For 29 of these problems, WPO is essential.¹⁰ In the competition, NaTT finished in the remarkable second place in the TRS standard category.

8. Conclusion

We introduced the weighted path order both as a reduction order and as a reduction pair. We presented several instances of WPO as reduction orders: $\text{WPO}(\mathcal{S}um)$ that subsumes KBO, $\text{WPO}(\mathcal{P}ol)$ that subsumes POLO and TKBO, $\text{WPO}(\mathcal{M}ax)$ that subsumes LPO, $\text{WPO}(\mathcal{M}Sum)$ that unifies KBO and LPO, and $\text{WPO}(\mathcal{M}Pol)$ that unifies all of them. Moreover, we applied partial status for WPO to obtain a reduction pair, and presented further refinements. We show

⁸<http://www.lsi.upc.edu/~albert/term.html>

⁹However, KBO does not contribute in this strategy.

¹⁰Due to the efficiency of our implementation, our tool proves 7 open problems in TPDB without using WPO.

that as a reduction pair, WPO subsumes KBO, LPO and TKBO with argument filters, POLO, and matrix interpretations. We also presented SMT encodings for these techniques. The orientability problems of $\text{WPO}(\text{Sum})$, $\text{WPO}(\text{Max})$ and $\text{WPO}(\text{MSum})$ are decidable, since they are reduced to satisfiability problems of linear integer arithmetic which is known to be decidable. Finally, we verified through experiments the significance of our work both as a reduction order and as a reduction pair. In order to keep the presentation simple, we did not present WPO with *multiset status*. Nonetheless, it is easy to define WPO with multiset status and verify that $\text{WPO}(\text{Max})$ with multiset status subsumes RPO.

We only considered a straightforward method for combining $\text{WPO}(\text{Pol})$ and $\text{WPO}(\text{Max})$ using ‘weight statuses’, and moreover heuristically fixed the weight status. We leave it for future work to search for other possible weight statuses, or to find more sophisticated combinations of max-polynomials such as $f_A(x, y, z) = x + \max(y, z)$, or even trying other algebras including *ordinal interpretations* [13, 49]. For efficiency, real arithmetic is also attractive to consider, since SMT for real arithmetic is often more efficient than for integer arithmetic. To this end, we will have to reconstruct the proof of well-foundedness of WPO, since our current proof relies on well-foundedness of the underlying order, which does not hold anymore for real numbers. Another obvious future work is to extend WPO for higher-order case. Since RPOLO has strength in its higher-order version [22], we expect their technique can be extended for WPO.

Acknowledgments. We are grateful to the anonymous reviewers for their careful inspections and comments that significantly improved the presentation of this paper. We thank Sarah Winkler and Aart Middeldorp for discussions at the early stages of this work. We thank Florian Frohn and Jürgen Giesl for their helps in experiments with AProVE, and Albert Rubio, Miquel Bofill and Cristina Borralleras for their helps in experiments with THOR. This work was supported by JSPS KAKENHI #24500012.

Appendix A. Omitted Proofs

In this appendix, we prove several properties that are needed for soundness of WPO. The first one is obvious from the definition.

Lemma 6. $\succ_{\text{WPO}} \subseteq \lesssim_{\text{WPO}}$. □

Using the above lemma, we show compatibility of \lesssim_{WPO} and \succ_{WPO} .

Lemma 7 (Compatibility). \succ_{WPO} is compatible with \lesssim_{WPO} .

Proof. Supposing $s \lesssim_{\text{WPO}} t \succ_{\text{WPO}} u$, we show $s \succ_{\text{WPO}} u$ by induction on $\langle |s|, |t|, |u| \rangle$. The other case, $s \succ_{\text{WPO}} t \lesssim_{\text{WPO}} u$, is analogous.

From the definition, it is obvious that t is of the form $g(\bar{t}_m)$ and moreover $s \gtrsim_A t \gtrsim_A u$. If $s >_A t$ or $t >_A u$, then we obtain $s \succ_{\text{WPO}} u$ by case (1). If $s \in \mathcal{V}$, then only case (2c) is applicable for $s \succ_{\text{WPO}} t$. In this case, $t \succ_{\text{WPO}} u$ is not possible, since $\sigma(g) = []$ and $g \not\prec_{\mathcal{F}} h$ for any h . Hence, s is of the form $f(\bar{s}_n)$. If $s_i \lesssim_{\text{WPO}} t$ for some $i \in \sigma(f)$, then by the induction hypothesis and Lemma 6, we get $s_i \lesssim_{\text{WPO}} u$ and hence case (2a) applies for $s \succ_{\text{WPO}} t$. Now suppose $s \succ_{\text{WPO}} t_j$ for every $j \in \sigma(g)$ and either $f >_{\mathcal{F}} g$ or $f \sim_{\mathcal{F}} g$ and $[\bar{s}_{\sigma(f)}] \lesssim_{\text{WPO}}^{\text{lex}} [\bar{t}_{\sigma(g)}]$ holds. There remain the following cases to consider for $t \succ_{\text{WPO}} u$:

- $t_j \succsim_{\text{WPO}} u$ for some $j \in \sigma(g)$. In this case, we have $s \succ_{\text{WPO}} t_j \succsim_{\text{WPO}} u$. Hence, by the induction hypothesis we conclude $s \succ_{\text{WPO}} u$.
- $u = h(\overline{u}_l)$ and $t \succ_{\text{WPO}} u_k$ for every $k \in \sigma(h)$. By the induction hypothesis, we obtain $s \succ_{\text{WPO}} u_k$. Moreover, if $f \succ_{\mathcal{F}} g$ or $g \succ_{\mathcal{F}} h$, then we get $f \succ_{\mathcal{F}} h$ and (2b-i) applies. Otherwise, we have $f \sim_{\mathcal{F}} g \sim_{\mathcal{F}} h$ and $[\overline{s}_{\sigma(f)}] \succ_{\text{WPO}}^{\text{lex}} [\overline{t}_{\sigma(g)}] \succ_{\text{WPO}}^{\text{lex}} [\overline{u}_{\sigma(h)}]$. From the induction hypothesis, we obtain $[\overline{s}_{\sigma(f)}] \succ_{\text{WPO}}^{\text{lex}} [\overline{u}_{\sigma(h)}]$. Hence (2b-ii) applies. \square

In order to prove well-foundedness of \succ_{WPO} , we define the set **SN** of *strongly normalizing* terms as follows: $s \in \text{SN}$ iff there exists no infinite reduction sequence $s \succ_{\text{WPO}} s_1 \succ_{\text{WPO}} s_2 \succ_{\text{WPO}} \dots$ beginning from s . In the next two lemmas, we prove that all terms are in **SN**.

Lemma 8. *Suppose that \mathcal{A} is weakly simple w.r.t. σ . If $s = f(\overline{s}_n)$ with $s_i \in \text{SN}$ for every $i \in \sigma(f)$, then $s \succ_{\text{WPO}} t$ implies $t \in \text{SN}$.*

Proof. We perform induction on $\langle s, f, [\overline{s}_{\sigma(f)}], |t| \rangle$ which is ordered by the lexicographic composition of $\succ_{\mathcal{A}}$, $\succ_{\mathcal{F}}$, $\succ_{\text{WPO}}^{\text{lex}}$ and $>$. Since the claim is obvious if $t \in \text{Var}$, we consider $t = g(t_1, \dots, t_m)$.

1. Suppose $s \succ_{\mathcal{A}} t$. First we show that $t_j \in \text{SN}$ for every $j \in \sigma(g)$. By the weak simplicity assumption, we have $s \succ_{\mathcal{A}} t \succ_{\mathcal{A}} t_j$ and hence $s \succ_{\text{WPO}} t_j$. Thus by the induction hypothesis on the fourth component, we obtain $t_j \in \text{SN}$. Now for arbitrary u s.t. $t \succ_{\text{WPO}} u$, the induction hypothesis on the first component yields $u \in \text{SN}$.
2. Suppose $s \succ_{\mathcal{A}} t$. There are two subcases to consider.
 - (a) Suppose $s_i \succsim_{\text{WPO}} t$ for some $i \in \sigma(f)$. Then by the assumption, we have $s_i \in \text{SN}$. Hence by Lemma 7, we obtain $t \in \text{SN}$.
 - (b) Suppose $s \succ_{\text{WPO}} t_j$ for every $j \in \sigma(g)$. Then by the induction hypothesis on the fourth component, we get $t_j \in \text{SN}$. Consider an arbitrary u s.t. $t \succ_{\text{WPO}} u$. Since we have either $f \succ_{\mathcal{F}} g$ or $f \succsim_{\mathcal{F}} g$ and $[\overline{s}_{\sigma(f)}] \succ_{\text{WPO}}^{\text{lex}} [\overline{t}_{\sigma(g)}]$, $\langle s, f, [\overline{s}_{\sigma(f)}], |t| \rangle$ is greater than $\langle t, g, [\overline{t}_{\sigma(g)}], |u| \rangle$ by the second or third component. Hence, the induction hypothesis yields $u \in \text{SN}$. \square

Lemma 9 (Well-foundedness). *If \mathcal{A} is weakly simple w.r.t. σ , then \succ_{WPO} is well-founded.*

Proof. Let us show $s \in \text{SN}$ for every term s by induction on $|s|$. The claim is trivial if $s \in \mathcal{V}$. Suppose $s = f(\overline{s}_n) \succ_{\text{WPO}} t$. By the induction hypothesis, we have $s_i \in \text{SN}$ for every $i \in \{1, \dots, n\}$. Hence by Lemma 8, we get $t \in \text{SN}$. \square

Now we prove that \succsim_{WPO} and \succ_{WPO} are quasi- and strict orders, resp.

Lemma 10 (Transitivity). *Both \succsim_{WPO} and \succ_{WPO} are transitive.*

Proof. Analogous to Lemma 7. \square

Lemma 11 ((Ir)reflexivity). *\succsim_{WPO} is reflexive and \succ_{WPO} is irreflexive.*

Proof. For arbitrary term s , $s \succsim_{\text{WPO}} s$ is easy by induction on $|s|$. Irreflexivity of \succ_{WPO} follows from Lemma 9. \square

Now the remaining properties required for a reduction pair are *stability* and *weak monotonicity*, which are shown below.

Lemma 12 (Stability). *Both \succsim_{WPO} and \succ_{WPO} are stable.*

Proof. Let $s = f(\bar{s}_n) \prec_{\text{WPO}} t$ and θ be an arbitrary substitution. We show $s\theta \prec_{\text{WPO}} t\theta$ by induction on $|s| + |t|$. The claim is obvious if $s \succ_{\mathcal{A}} t$. Otherwise, we have $s \succsim_{\mathcal{A}} t$ and obviously $s\theta \succsim_{\mathcal{A}} t\theta$. The remaining cases are as follows:

- Suppose $s_i \prec_{\text{WPO}} t$ for some $i \in \sigma(f)$. By the induction hypothesis, we get $s_i\theta \prec_{\text{WPO}} t\theta$. Hence, case (2a) applies for $s\theta \prec_{\text{WPO}} t\theta$.
- Suppose $t = g(\bar{t}_m)$ and $s \succ_{\text{WPO}} t_j$ for all $j \in \sigma(g)$. By the induction hypothesis, we get $s\theta \succ_{\text{WPO}} t_j\theta$. The claim is obvious if $f \succ_{\mathcal{F}} g$. If $f \sim_{\mathcal{F}} g$ and $[\bar{s}_{\sigma(f)}] \prec_{\text{WPO}}^{\text{lex}} [\bar{t}_{\sigma(g)}]$, then by the induction hypothesis we get

$$[\bar{s}_{\sigma(f)}\theta] \prec_{\text{WPO}}^{\text{lex}} [\bar{t}_{\sigma(g)}\theta]$$

Hence, case (2b-ii) applies for $s\theta \prec_{\text{WPO}} t\theta$.

- If $s \prec_{\text{WPO}} t$ is derived by either case (2c) or (2d), then the claim follows from Proposition 2 or 3. \square

Lemma 13 (Weak monotonicity). *If \mathcal{A} is weakly monotone and weakly simple w.r.t. σ , then \succsim_{WPO} is monotone.*

Proof. Suppose $s_i \prec_{\text{WPO}} s'_i$ and let us show $s = f(\dots, s_i, \dots) \prec_{\text{WPO}} f(\dots, s'_i, \dots) = s'$. Since $s_i \succsim_{\mathcal{A}} s'_i$, we have $s \succsim_{\mathcal{A}} s'$ by the weak monotonicity of \mathcal{A} .

If $i \notin \sigma(f)$, then we have $[\bar{s}_{\sigma(f)}] = [\bar{s}'_{\sigma(f)}]$. Otherwise, by the weak simplicity assumption we have $s \succsim_{\mathcal{A}} s_j$ for every $j \in \sigma(f)$, and thus $s \succ_{\text{WPO}} s_j$ by case (2a) of Definition 10. Hence case (2b-ii) applies for $s \prec_{\text{WPO}} s'$. \square

The above results conclude Theorem 22. Now we prove Theorem 13. Most of the required properties have already been obtained above, except for the following two:

Lemma 14 (Strict monotonicity). *If \mathcal{A} is weakly monotone and weakly simple, then \succ_{WPO} is monotone.*

Proof. Suppose $s_i \succ_{\text{WPO}} s'_i$ and let us show $s = f(\dots, s_i, \dots) \succ_{\text{WPO}} f(\dots, s'_i, \dots) = s'$. Since $s_i \succ_{\mathcal{A}} s'_i$, we have $s \succ_{\mathcal{A}} s'$ by the weak monotonicity of \mathcal{A} . By the weak simplicity of \mathcal{A} , we have $s \succ_{\mathcal{A}} s_j$ for every j , and thus $s \succ_{\text{WPO}} s_j$ by case (2a) of Definition 5. Hence case (2b-ii) applies for $s \succ_{\text{WPO}} s'$. \square

Lemma 15 (Subterm property). *If \mathcal{A} is weakly simple, then \succ_{WPO} has the subterm property.*

Proof. By the assumption, $f(\dots, s_i, \dots) \succ_{\mathcal{A}} s_i$ and hence $f(\dots, s_i, \dots) \succ_{\text{WPO}} s_i$ by case (2a). \square

References

- [1] J. Giesl, P. Schneider-Kamp, R. Thiemann, AProVE 1.2: Automatic termination proofs in the dependency pair framework, in: Proceedings of 3rd International Joint Conference on Automated Reasoning (IJCAR '06), Volume 4130 of LNAI, 2006, pp. 281–286.
- [2] M. Korp, C. Sternagel, H. Zankl, A. Middeldorp, Tyrolean Termination Tool 2, in: Proceedings of 20th International Conference on Rewriting Techniques and Applications (RTA '09), Volume 5595 of LNCS, 2009, pp. 295–304.
- [3] S. Kamin, J.-J. Lévy, Two generalizations of the recursive path ordering, 1980. Unpublished note.
- [4] N. Dershowitz, Orderings for term-rewriting systems, *Theor. Comput. Sci.* 17 (1982) 279–301.
- [5] P. Lescanne, Computer experiments with the REVE term rewriting system generator, in: Proceedings of 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL '83), 1983, pp. 99–108.
- [6] M. Codish, J. Giesl, P. Schneider-Kamp, R. Thiemann, SAT solving for termination proofs with recursive path orders and dependency pairs, *J. Autom. Reasoning* 49 (2012) 53–93.
- [7] D. Knuth, P. Bendix, Simple word problems in universal algebras, in: *Computational Problems in Abstract Algebra*, Pergamon Press, New York, 1970, pp. 263–297.
- [8] K. Korovin, A. Voronkov, Orienting rewrite rules with the Knuth-Bendix order, *Inf. Comput.* 183 (2003) 165–186.
- [9] H. Zankl, N. Hirokawa, A. Middeldorp, KBO orientability, *J. Autom. Reasoning* 43 (2009) 173–201.
- [10] A. Middeldorp, H. Zantema, Simple termination of rewrite systems, *Theor. Comput. Sci.* 175 (1997) 127–158.
- [11] M. Ludwig, U. Waldmann, An extension of the Knuth-Bendix ordering with LPO-like properties, in: Proceedings of 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '07), Volume 4790 of LNAI, 2007, pp. 348–362.
- [12] L. Kovács, G. Moser, A. Voronkov, On transfinite Knuth-Bendix orders, in: Proceedings of 23rd International Conference on Automated Deduction (CADE '11), Volume 6803 of LNAI, 2011, pp. 384–399.
- [13] S. Winkler, H. Zankl, A. Middeldorp, Ordinals and Knuth-Bendix orders, in: Proceedings of 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '12), Volume 7180 of LNCS ARCoSS, 2012, pp. 420–434.
- [14] D. Lankford, Canonical algebraic simplification in computational logic, Technical Report ATP-25, University of Texas, 1975.

- [15] H. Zantema, The termination hierarchy for term rewriting, *Appl. Algebr. Eng. Comm. Compt.* 12 (2001) 3–19.
- [16] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, H. Zankl, Maximal termination, in: *Proceedings of 19th International Conference on Rewriting Techniques and Applications (RTA '08)*, Volume 5117 of LNCS, 2008, pp. 110–125.
- [17] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, H. Zankl, SAT solving for termination analysis with polynomial interpretations, in: *Proceedings of 10th Theory and Applications of Satisfiability Testing (SAT '07)*, Volume 4501 of LNCS, 2007, pp. 340–354.
- [18] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, *Theor. Compt. Sci.* 236 (2000) 133–178.
- [19] N. Hirokawa, A. Middeldorp, Automating the dependency pair method, *Inf. Comput.* 199 (2005) 172–199.
- [20] J. Giesl, R. Thiemann, P. Schneider-Kamp, S. Falke, Mechanizing and improving dependency pairs, *J. Autom. Reasoning* 37 (2006) 155–203.
- [21] J. Endrullis, J. Waldmann, H. Zantema, Matrix interpretations for proving termination of term rewriting, *J. Autom. Reasoning* 40 (2008) 195–220.
- [22] M. Bofill, C. Borralleras, E. Rodríguez-Carbonell, A. Rubio, The recursive path and polynomial ordering for first-order and higher-order terms, *J. Logic Comput.* 23 (2013) 263–305.
- [23] C. Borralleras, M. Ferreira, A. Rubio, Complete monotonic semantic path orderings, in: *Proceedings of 17th International Conference on Automated Deduction (CADE '00)*, Volume 1831 of LNCS, 2000, pp. 346–364.
- [24] N. Dershowitz, C. Hoot, Natural termination, *Theor. Compt. Sci.* 142 (1995) 179–207.
- [25] A. Geser, An improved general path order, *Appl. Algebr. Eng. Comm. Compt.* 7 (1996) 469–511.
- [26] A. Yamada, K. Kusakari, T. Sakabe, Partial status for KBO, in: *Proceedings of 13th International Workshop on Termination (WST '13)*, 2013, pp. 74–78.
- [27] TPDB, The termination problem data base, version 8.0.6, 2013. URL: <http://termination-portal.org/wiki/TPDB>.
- [28] A. Yamada, K. Kusakari, T. Sakabe, Unifying the Knuth-Bendix, recursive path and polynomial orders, in: *Proceedings of 15th International Symposium on Principles and Practice of Declarative Programming (PPDP '13)*, 2013, pp. 181–192.
- [29] F. Baader, T. Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.

- [30] TeReSe, Term Rewriting Systems, Volume 55 of Cambridge Tracts Theoret. Comput. Sci., Cambridge University Press, 2003.
- [31] H. Zantema, Termination of term rewriting: interpretation and type elimination, *J. Symb. Comput.* 17 (1994) 23–50.
- [32] Z. Manna, S. Ness, On the termination of Markov algorithms, in: *Proceedings of the 3rd Hawaii International Conference on System Science*, 1970, pp. 789–792.
- [33] J. Steinbach, Extensions and comparison of simplification orders, in: *Proceedings of 3rd International Conference on Rewriting Techniques and Applications (RTA '89)*, Volume 355 of LNCS, 1989, pp. 434–448.
- [34] J. Giesl, R. Thiemann, P. Schneider-Kamp, The dependency pair framework: Combining techniques for automated termination proofs, in: *Proceedings of 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '04)*, Volume 3452 of LNAI, 2004, pp. 75–90.
- [35] K. Kusakari, M. Nakamura, Y. Toyama, Argument filtering transformation, in: *Proceedings of 1st International Symposium on Principles and Practice of Declarative Programming (PPDP '99)*, Volume 1702 of LNCS, 1999, pp. 47–61.
- [36] A. Koprowski, J. Waldmann, Max/plus tree automata for termination of term rewriting, *Acta Cybern.* 19 (2009) 357–392.
- [37] C. Sternagel, R. Thiemann, Formalizing Knuth-Bendix orders and Knuth-Bendix completion, in: *Proceedings of 24th International Conference on Rewriting Techniques and Applications (RTA '13)*, Volume 21 of LIPIcs, 2013, pp. 287–302.
- [38] R. Thiemann, G. Allais, J. Nagele, On the formalization of termination techniques based on multiset orderings, in: *Proceedings of 23rd International Conference on Rewriting Techniques and Applications (RTA '12)*, Volume 15 of LIPIcs, 2012, pp. 339–354.
- [39] K. Korovin, A. Voronkov, An AC-compatible Knuth-Bendix order, in: *Proceedings of 19th International Conference on Automated Deduction (CADE '03)*, Volume 2741 of LNAI, 2003, pp. 47–59.
- [40] A. M. Ben-Amram, M. Codish, A SAT-based approach to size change termination with global ranking functions, in: *Proceedings of 14th Tools and Algorithms for the Construction and Analysis of Systems (TACAS '08)*, Volume 4963 of LNCS, 2008, pp. 218–232.
- [41] J. Giesl, R. Thiemann, P. Schneider-Kamp, Proving and disproving termination of higher-order functions, in: *Proceedings of 5th International Symposium on Frontiers of Combining Systems (FroCoS '05)*, Volume 3717 of LNAI, 2005, pp. 216–231.

- [42] M. Codish, P. Schneider-Kamp, V. Lagoon, R. Thiemann, J. Giesl, SAT solving for argument filterings, in: Proceedings of 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'06), Volume 4246 of LNCS, 2006, pp. 30–44.
- [43] TermComp, The termination competition, 2013. URL: <http://termcomp.uibk.ac.at/termcomp/>.
- [44] H. Zankl, A. Middeldorp, Satisfiability of non-linear (ir)rational arithmetic, in: Proceedings of 16th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'10), Volume 6355 of LNAI, 2010, pp. 481–500.
- [45] C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, A. Rubio, SAT modulo linear arithmetic for solving polynomial constraints, *J. Autom. Reasoning* 48 (2012) 107–131.
- [46] N. Hirokawa, A. Middeldorp, H. Zankl, Uncurrying for termination and complexity, *J. Autom. Reasoning* 50 (2013) 279–315.
- [47] C. Sternagel, R. Thiemann, Generalized and formalized uncurrying, in: Proceedings of 8th International Symposium on Frontiers of Combining Systems (FroCoS'11), Volume 6989 of LNAI, 2011, pp. 243–258.
- [48] A. Yamada, K. Kusakari, T. Sakabe, Nagoya Termination Tool, in: Proceedings of Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications (RTA-TLCA '14), LNCS, 2014. To appear.
- [49] S. Winkler, H. Zankl, A. Middeldorp, Beyond Peano arithmetic – automatically proving termination of the Goodstein sequence, in: Proceedings of 24th International Conference on Rewriting Techniques and Applications (RTA'13), Volume 21 of LIPIcs, 2013, pp. 335–351.